Transport Services (tsv)
Internet-Draft
Intended status: Experimental
Expires: April 21, 2016

K. De Schepper
Bell Labs
B. Briscoe, Ed.
Simula Research Lab
I. Tsang
Bell Labs

October 19, 2015

Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay

draft-briscoe-tsvwg-ecn-l4s-id-00

Abstract

This specification defines the identifier to be used on IP packets for a new network service called low latency, low loss and scalable throughput (L4S). It is similar to the original (or 'Classic') Explicit Congestion Notification (ECN). 'Classic' ECN marking was required to be equivalent to a drop, both when applied in the network and when responded to by a transport. Unlike 'Classic' ECN marking, the network applies the L4S identifier more immediately and more aggressively than drop, and the transport response to each mark is reduced and smoothed relative to that for drop. The two changes counterbalance each other so that the bit-rate of an L4S flow will be roughly the same as a 'Classic' flow under the same conditions. However, the much more frequent control signals and the finer responses to them result in ultra-low queuing delay without compromising link utilization, even during high load. Examples of new active queue management (AQM) marking algorithms and examples of new transports (whether TCP-like or real-time) are specified separately. The new L4S identifier is the key piece that enables them to interwork and distinguishes them from 'Classic' traffic. It gives an incremental migration path so that existing 'Classic' TCP traffic will be no worse off, but it can be prevented from degrading the ultra-low delay and loss of the new scalable transports.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This specification defines the identifier to be used on IP packets for a new network service called low latency, low loss and scalable throughput (L4S). It is similar to the original (or 'Classic') Explicit Congestion Notification (ECN). 'Classic' ECN marking was required to be equivalent to a drop, both when applied in the network and when responded to by a transport. Unlike 'Classic' ECN marking, the network applies the L4S identifier more immediately and more aggressively than drop, and the transport response to each mark is reduced and smoothed relative to that for drop. The two changes counterbalance each other so that the bit-rate of an L4S flow will be roughly the same as a 'Classic' flow under the same conditions. However, the much more frequent control signals and the finer responses to them result in ultra-low queuing delay without compromising link utilization, even during high load.

An example of an active queue management (AQM) marking algorithm that enables the L4S service is the DualQ Coupled AQM defined in a complementary specification [I-D.briscoe-aqm-dualq-coupled]. An example of a scalable transport that would enable the L4S service is Data Centre TCP (DCTCP), which until now has been applicable solely to controlled environments like data centres [I-D.bensley-tcpm-dctcp], because it is too aggressive to co-exist with existing TCP. However, AQMs like DualQ Coupled enable scalable transports like DCTCP to co-exist with existing traffic, each getting roughly the same flow rate when they compete under similar conditions.

The new L4S identifier is the key piece that enables these two parts to interwork and distinguishes them from 'Classic' traffic. It gives an incremental migration path so that existing 'Classic' TCP traffic will be no worse off, but it can be prevented from degrading the ultra-low delay and loss of the new scalable transports. The performance improvement is so great that it is hoped it will motivate initial deployment of the separate parts of this system.

1.1. Problem

Latency is becoming the critical performance factor for many (most?) applications on the public Internet, e.g. Web, voice, conversational video, gaming and finance apps. In the developed world, further increases in access network bit-rate offer diminishing returns, whereas latency is still a multi-faceted problem. In the last decade or so, much has been done to reduce propagation time by placing

caches or servers closer to users. However, queuing remains a major component of latency.

The Diffserv architecture provides Expedited Forwarding [RFC3246], so that low latency traffic can jump the queue of other traffic. However, on access links dedicated to individual sites (homes, small enterprises or mobile devices), often all traffic at any one time will be latency-sensitive. Then Diffserv is of little use. Instead, we need to remove the causes of any unnecessary delay.

The bufferbloat project has shown that excessively-large buffering (‘bufferbloat’) has been introducing significantly more delay than the underlying propagation time. These delays appear only intermittently--only when a capacity-seeking (e.g. TCP) flow is long enough for the queue to fill the buffer, making every packet in other flows sharing the buffer sit through the queue.

Active queue management (AQM) was originally developed to solve this problem (and others). Unlike Diffserv, which gives low latency to some traffic at the expense of others, AQM controls latency for _all_ traffic in a class. In general, AQMs introduce an increasing level of discard from the buffer the longer the queue persists above a shallow threshold. This gives sufficient signals to capacity-seeking (aka. greedy) flows to keep the buffer empty for its intended purpose: absorbing bursts. However, RED [RFC2309] and other algorithms from the 1990s were sensitive to their configuration and hard to set correctly. So, AQM was not widely deployed. More recent state-of-the-art AQMs, e.g. fq_CoDel [I-D.ietf-aqm-fq-codel], PIE [I-D.ietf-aqm-pie], Adaptive RED [ARED01], define the threshold in time not bytes, so it is invariant for different link rates.

Latency is not our only concern: It was known when TCP was first developed that it would not scale to high bandwidth-delay products. Given regular broadband bit-rates over WAN distances are already [RFC3649] beyond the scaling range of ‘classic’ TCP Reno, ‘less unscalable’ Cubic [I-D.zimmermann-tcpm-cubic] and Compound [I-D.sridharan-tcpm-ctcp] variants of TCP have been successfully deployed. However, these are now approaching their scaling limits. Unfortunately, fully scalable TCPs such as DCTCP [I-D.bensley-tcpm-dctcp] cause ‘classic’ TCP to starve itself, which is why they have been confined to private data centres or research testbeds (until now).

It turns out that a TCP algorithm like DCTCP that solves TCP’s scalability problem also solves the latency problem, because the finer sawteeth cause very little queuing delay. A supporting paper [DCttH15] gives the full explanation of why the design solves both the latency and the scaling problems, both in plain English and in

more precise mathematical form. THe explanation is summarised without the maths in [I-D.briscoe-aqm-dualq-coupled].

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

Classic service: The ‘Classic’ service is intended for all the behaviours that currently co-exist with TCP Reno (TCP Cubic, Compound, SCTP, etc).

Low-Latency, Low-Loss and Scalable (L4S): The ‘L4S’ service is intended for traffic from scalable TCP algorithms such as Data Centre TCP. But it is also more general--it will allow a set of congestion controls with similar scaling properties to DCTCP (e.g. Relentless [Mathis09]) to evolve.

Both Classic and L4S services can cope with a proportion of unresponsive or less-responsive traffic as well (e.g. DNS, VoIP, etc).

Classic ECN: The original Explicit Congestion Notification (ECN) protocol [RFC3168].

1.3. Scope

.The new L4S identifier defined in this specification is applicable for IPv4 and IPv6 packets (as for classic ECN [RFC3168]). It is applicable for the unicast, multicast and anycast forwarding modes. It is an orthogonal packet classification to Differentiated Services (Diffserv [RFC2474]), therefore it can be applied to any packet in any Diffserv traffic class. However, as with classic ECN, any particular forwarding node might not implement an active queue management algorithm in all its DIffserv queues.

This document is intended for experimental status, so it does not update any standards track RFCs. If the experiment is successful and this document proceeds to the standards track, it would be expected to update the specification of ECN in IP and in TCP [RFC3168]. For packets carrying the L4S identifier, it would update both the network’s ECN marking behaviour and the TCP response to ECN feedback, making them distinct from the behaviours for drop. It would also update the specification of ECN in RTP over UDP [RFC6679] {ToDo: DCCP

and SCTP refs}. Finally, it would also obsolete the experimental ECN nonce [RFC3540].

2. L4S Packet Identifier

2.1. L4S Packet Identification Requirements

Ideally, the identifier for packets using the Low Latency, Low Loss, Scalable throughput (L4S) service ought to meet the following requirements:

- o it SHOULD survive end-to-end between source and destination applications: across the boundary between host and network, between interconnected networks, and through middleboxes;

- o it SHOULD be common to IPv4 and IPv6;

- o it SHOULD be incrementally deployable;

- o it SHOULD enable an AQM to classify packets encapsulated by outer IP or lower-layer headers;

- o it SHOULD consume minimal extra codepoints;

- o it SHOULD not lead to some packets of a transport-layer flow being served by a different queue from others.

It is recognised that the chosen identifier is unlikely to satisfy all these requirements, particularly given the limited space left in the IP header. Therefore a compromise will be necessary, which is why all the requirements are expressed with the word 'SHOULD' not 'MUST'. Appendix A discusses the pros and cons of the compromises made in various competing identification schemes. The chosen scheme is defined in Section 2.2 below.

Whether the identifier would be recoverable if the experiment failed is a factor that could be taken into account. However, this has not been made a requirement, because that would favour schemes that would be easier to fail, rather than those more likely to succeed.

2.2. L4S Packet Identification

The L4S treatment is an alternative packet marking treatment [RFC4774] to the classic ECN treatment [RFC3168]. Like classic ECN, it identifies the marking treatment that network nodes are expected to apply to L4S packets, and it identifies packets that are expected to have been sent from hosts applying a broad type of behaviour, termed L4S congestion control.

For a packet to receive L4S treatment as it is forwarded, the sender MUST set the ECN field in the IP header (v4 or v6) to the ECT(1) codepoint.

A network node that implements the L4S service MUST classify arriving ECT(1) packets for L4S treatment and it SHOULD classify arriving CE packets for L4S treatment as well. Section 2.3 describes an exception to this latter rule.

The L4S AQM treatment follows similar codepoint transition rules to those in RFC 3168. Specifically, the ECT(1) codepoint MUST NOT be changed to any other codepoint than CE, and CE MUST NOT be changed to any other codepoint. An ECT(1) packet is classified as ECN-capable and, if congestion increases, an L4S AQM algorithm will set the ECN marking of an increasing proportion of packets to CE, otherwise forwarding packets unchanged as ECT(1). The L4S marking treatment is defined in Section 2.4. Under persistent overload conditions, the AQM will follow RFC 3168 and turn off ECN marking, using drop as a congestion signal until the overload episode has subsided.

The L4S treatment is the default for ECT(1) packets in all Diffserv Classes [RFC4774].

For backward compatibility, a network node that implements the L4S treatment MUST also implement a classic AQM treatment. It MUST classify arriving ECT(0) and Not-ECT packets for treatment by the Classic AQM. Classic treatment means that the AQM will mark ECT(0) packets under the same conditions as it would drop Not-ECT packets [RFC3168].

2.3. L4S Packet Identification with Transport-Layer Awareness

To implement the L4S treatment, a network node does not need to identify transport-layer flows. Nonetheless, if a network node is capable of identifying transport-layer flows, it SHOULD classify CE packets for classic ECN [RFC3168] treatment if the most recent ECT packet in the same flow was ECT(0). If a network node does not identify transport-layer flows, or if the most recent ECT packet was ECT(1), it MUST classify CE packets for L4S treatment.

Only the most recent ECT packet of a flow is used to classify a CE packet, because a sender might have to switch from sending ECT(1) (L4S) packets to sending ECT(0) (Classic) packets, or back again, in the middle of a transport-layer flow. Such a switch-over is likely to be very rare, but It could be necessary if the path bottleneck moves from a network node that supports L4S to one that only supports Classic ECN. Such a change ought to be detectable from the change in RTT variation.

2.4. The Meaning of CE Relative to Drop

The likelihood that an AQM drops a Not-ECT Classic packet MUST be proportional to the square of the likelihood that it would have marked it if it had been an L4S packet. The constant of proportionality does not have to be standardised for interoperability, but a value of 1 is RECOMMENDED.

[I-D.briscoe-aqm-dualq-coupled].specifies the essential aspects of an L4S AQM, as well as recommending other aspects. It gives an example implementation in an appendix.

The term ’likelihood’ is used above to allow for marking and dropping to be either probabilistic or deterministic. This example AQM in [I-D.briscoe-aqm-dualq-coupled] drops and marks probabilistically, so the drop probability is arranged to be the square of the marking probability. Nonetheless, an alternative AQM that dropped and marked deterministically would be valid, as long as the dropping frequency was proportional to the square of the marking frequency.

Note that, contrary to RFC 3168, an AQM implementing the L4S and Classic treatments does not mark an ECT(1) packet under the same conditions that it would have dropped a Not-ECT packet. However, it does mark and ECT(0) packet under the same conditions that it would have dropped a Not-ECT packet.

3. IANA Considerations

This specification contains no IANA considerations.

{ToDo: If this specification becomes and experimental RFC, should IANA be asked to update <http://www.iana.org/assignments/ipv4-tos-byte/ipv4-tos-byte.xhtml#ipv4-tos-byte-1> so that the reference for the specification of ECT(1) points to this document, and CE points to both RFC3168 and this document? I think not, because this experimental specification will not update RFC3168, which is standards track.}

4. Security Considerations

Two approaches to assure the integrity of signals using the new identifer are introduced in Appendix B.1.

5. Acknowledgements

Thanks to Richard Scheffenegger, John Leslie, David Taeht, Jonathan Morton, Gorry Fairhurst, Michael Welzl, Mikael Abrahamsson and Andrew McGregor for the discussions that led to this specification.

The authors' contributions are part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed here are solely those of the authors.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC4774] Floyd, S., "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field", BCP 124, RFC 4774, DOI 10.17487/RFC4774, November 2006, <http://www.rfc-editor.org/info/rfc4774>.

[RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <http://www.rfc-editor.org/info/rfc6679>.

6.2. Informative References

[ARED01] Floyd, S., Gummadi, R., and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management", ACIRI Technical Report , August 2001, <http://www.icir.org/floyd/red.html>.

[DCttH15] De Schepper, K., Bondarenko, O., Briscoe, B., and I. Tsang, "'Data Centre to the Home': Ultra-Low Latency for All", 2015, <http://www.bobbriscoe.net/projects/latency/dctth_preprint.pdf>.

(Under submission)

[I-D.bensley-tcpm-dctcp] Bensley, S., Eggert, L., Thaler, D., Balasubramanian, P., and G. Judd, "Microsoft's Datacenter TCP (DCTCP): TCP Congestion Control for Datacenters", draft-bensley-tcpm-dctcp-05 (work in progress), July 2015.

[I-D.briscoe-aqm-dualq-coupled]
Schepper, K., Briscoe, B., Bondarenko, O., and i. ing-jyh.tsang@alcatel-lucent.com, "DualQ Coupled AQM for Low Latency, Low Loss and Scalable Throughput", draft-briscoe-aqm-dualq-coupled-00 (work in progress), August 2015.

[I-D.ietf-aqm-fq-codel]
Hoeiland-Joergensen, T., McKenney, P., dave.taht@gmail.com, d., Gettys, J., and E. Dumazet, "FlowQueue-Codel", draft-ietf-aqm-fq-codel-01 (work in progress), July 2015.

[I-D.ietf-aqm-pie]
Pan, R., Natarajan, P., and F. Baker, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem", draft-ietf-aqm-pie-02 (work in progress), August 2015.

[I-D.ietf-conex-abstract-mech]
Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism and Requirements", draft-ietf-conex-abstract-mech-13 (work in progress), October 2014.

[I-D.ietf-tcpm-accecn-reqs]
Kuehlewind, M., Scheffenegger, R., and B. Briscoe, "Problem Statement and Requirements for a More Accurate ECN Feedback", draft-ietf-tcpm-accecn-reqs-08 (work in progress), March 2015.

[I-D.ietf-tsvwg-ecn-encap-guidelines]
Briscoe, B., Kaippallimalil, J., and P. Thaler, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP", draft-ietf-tsvwg-ecn-encap-guidelines-04 (work in progress), October 2015.

[I-D.moncaster-tcpm-rcv-cheat]
Moncaster, T., Briscoe, B., and A. Jacquet, "A TCP Test to Allow Senders to Identify Receiver Non-Compliance", draft-moncaster-tcpm-rcv-cheat-03 (work in progress), July 2014.

[I-D.sridharan-tcpm-ctcp]
Sridharan, M., Tan, K., Bansal, D., and D. Thaler, "Compound TCP: A New TCP Congestion Control for High-Speed and Long Distance Networks", draft-sridharan-tcpm-ctcp-02 (work in progress), November 2008.

[I-D.zimmermann-tcpm-cubic]
Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", draft-zimmermann-tcpm-cubic-01 (work in progress), April 2015.

[Mathis09]
Mathis, M., "Relentless Congestion Control", PFLDNeT'09 , May 2009, <http://www.hpcc.jp/pfldnet2009/Program_files/1569198525.pdf>.

[QV] Briscoe, B. and P. Hurtig, "Up to Speed with Queue View", RITE Technical Report , August 2015, <TBA>.

[RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, DOI 10.17487/RFC2309, April 1998, <http://www.rfc-editor.org/info/rfc2309>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <http://www.rfc-editor.org/info/rfc2474>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <http://www.rfc-editor.org/info/rfc2983>.

[RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, DOI 10.17487/RFC3246, March 2002, <http://www.rfc-editor.org/info/rfc3246>.

[RFC3540] Spring, N., Wetherall, D., and D. Ely, "Robust Explicit Congestion Notification (ECN) Signaling with Nonces", RFC 3540, DOI 10.17487/RFC3540, June 2003, <http://www.rfc-editor.org/info/rfc3540>.

[RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, DOI 10.17487/RFC3649, December 2003, <http://www.rfc-editor.org/info/rfc3649>.

[RFC5562] Kuzmanovic, A., Mondal, A., Floyd, S., and K. Ramakrishnan, "Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets", RFC 5562, DOI 10.17487/RFC5562, June 2009, <http://www.rfc-editor.org/info/rfc5562>.

[RFC6077] Papadimitriou, D., Ed., Welzl, M., Scharf, M., and B. Briscoe, "Open Research Issues in Internet Congestion Control", RFC 6077, DOI 10.17487/RFC6077, February 2011, <http://www.rfc-editor.org/info/rfc6077>.

[RFC6660] Briscoe, B., Moncaster, T., and M. Menth, "Encoding Three Pre-Congestion Notification (PCN) States in the IP Header Using a Single Diffserv Codepoint (DSCP)", RFC 6660, DOI 10.17487/RFC6660, July 2012, <http://www.rfc-editor.org/info/rfc6660>.

[VCP] Xia, Y., Subramanian, L., Stoica, I., and S. Kalyanaraman, "One more bit is enough", Proc. SIGCOMM'05, ACM CCR 35(4)37--48, 2005, <http://doi.acm.org/10.1145/1080091.1080098>.

Appendix A. Alternative Identifiers

This appendix is informative, not normative. It records the pros and cons of various alternative ways to identify L4S packets to record the rationale for the choice of ECT(1) (Appendix A.1) as the L4S identifier. At the end, Appendix A.6 summarises the distinguishing features of the leading alternatives,.It is intended to supplement, not replace the detailed text.

The leading solutions all use the ECN field, sometimes in combination with the Diffserv field. Both the ECN and Diffserv fields have the additional advantage that they are no different in either IPv4 or IPv6. A couple of alternatives that use other fields are mentioned at the end, but it is quickly explained why they are not serious contenders.

A.1. ECT(1) and CE codepoints

Definition:

Packets with ECT(1) and conditionally packets with CE would signify L4S semantics as an alternative to the semantics of classic ECN [RFC3168], specifically:

* The ECT(1) codepoint would signify that the packet was sent by an L4S-capable sender. Successful negotiation of accurate ECN

(AccECN) feedback [I-D.ietf-tcpm-accecn-reqs] is a pre-requisite for a sender to send L4S packets, therefore ECT(1) in turn signifies that both endpoints support AccECN;

* Given shortage of codepoints, both L4S and classic ECN sides of an AQM would have to use the same CE codepoint to indicate that a packet had experienced congestion. If a packet that had already been marked CE in an upstream buffer arrived at a subsequent AQM, this AQM would then have to guess whether to classify CE packets as L4S or classic ECN. Choosing the L4S treatment would be a safer choice, because then a few classic packets might arrive early, rather than a few L4S packets arriving late;

* Additional information might be available if the classifier were transport-aware. Then it could classify a CE packet for classic ECN treatment if the most recent ECT packet in the same flow had been marked ECT(0). However, the L4S service should not need tranport-layer awareness;

Cons:

Consumes the last ECN codepoint: The L4S service is intended to supersede the service provided by classic ECN, therefore using ECT(1) to identify L4S packets could ultimately mean that the ECT(0) codepoint was ‘wasted’ purely to distinguish one form of ECN from its successor;

ECN hard in some lower layers: It is not always possible to support ECN in an AQM acting in a buffer below the IP layer [I-D.ietf-tsvwg-ecn-encap-guidelines]. In such cases, the L4S service would have to drop rather than mark frames even though they might contain an ECN-capable packet. However, such cases would be unusual.

Risk of reordering classic CE packets: Having to classify all CE packets as L4S risks some classic CE packets arriving early, which is a form of reordering. Reordering can cause the TCP sender to retransmit spuriously. However, one or two packets delivered early does not cause any spurious retransmissions because the subsequent packets continue to move the cumulative acknowledgement boundary forwards. Anyway, even the risk of reordering would be low, because: i) it is quite unusual to experience more than one bottleneck queue on a path; ii) even then, reordering would only occur if there was simultaneous mixing of classic and L4S traffic, which would be more unlikely in an access link, which is where most bottlenecks are located; iii) even then, spurious retransmissions would only occur if a contiguous sequence of three

or more classic CE packets from one bottleneck arrived at the next, which should in itself happen very rarely with a good AQM. The risk would be completely eliminated in AQMs that were transport-aware (but they should not need to be);

Non-L4S service for control packets: The classic ECN RFCs [RFC3168] and [RFC5562] require a sender to clear the ECN field to Not-ECT for retransmissions and certain control packets specifically pure ACKs, window probes and SYNs. When L4S packets are classified by the ECN field alone, these control packets would not be classified into an L4S queue, and could therefore be delayed relative to the other packets in the flow. This would not cause re-ordering (because retransmissions are already out of order, and the control packets carry no data). However, it would make critical control packets more vulnerable to loss and delay. {ToDo: Discuss the likelihood that all these packets might be made ECN-capable in future.}

Pros:

Should work e2e: The ECN field generally works end-to-end across the Internet. Unlike the DSCP, the setting of the ECN field is at least forwarded unchanged by networks that do not support ECN, and networks rarely clear it to zero;

Should work in tunnels: Unlike Diffserv, ECN is defined to always work across tunnels. However, tunnels do not always implement ECN processing as they should do, particularly because IPsec tunnels were defined differently for a few years.

Could migrate to one codepoint: If all classic ECN senders eventually evolve to use the L4S service, the ECT(0) codepoint could be reused for some future purpose, but only once use of ECT(0) packets had reduced to zero, or near-zero, which might never happen.

A.2. ECN Plus a Diffserv Codepoint (DSCP)

Definition:

For packets with a defined DSCP, all codepoints of the ECN field (except Not-ECT) would signify alternative L4S semantics to those for classic ECN [RFC3168], specifically:

* The L4S DSCP would signifiy that the packet came from an L4S-capable sender;

* ECT(0) and ECT(1) would both signify that the packet was travelling between transport endpoints that were both ECN-capable and supported accurate ECN feedback [I-D.ietf-tcpm-accecn-reqs];

* CE would signify that the packet had been marked by an AQM implementing the L4S service.

Use of a DSCP is the only approach for alternative ECN semantics given as an example in [RFC4774]. However, it was perhaps considered more for controlled environments than new end-to-end services;

Cons:

Consumes DSCP pairs: A DSCP is obviously not orthogonal to Diffserv. Therefore, wherever the L4S service is applied to multiple Diffserv scheduling behaviours, it would be necessary to replace each DSCP with a pair of DSCPs.

Uses critical lower-layer header space: The resulting increased number of DSCPs might be hard to support for some lower layer technologies, e.g. 802.1p and MPLS both offer only 3-bits for a maximum of 8 traffic class identifiers. Although L4S should reduce and possibly remove the need for some DSCPs intended for differentiated queuing delay, it will not remove the need for Diffserv entirely, because Diffserv is also used to allocate bandwidth, e.g. by prioritising some classes of traffic over others when traffic exceeds available capacity.

Not end-to-end (host-network): Very few networks honour a DSCP set by a host. Typically a network will zero (bleach) the Diffserv field from all hosts. Sometimes networks will attempt to identify applications by some form of packet inspection and, based on network policy, they will set the DSCP considered appropriate for the identified application. Network-based application identification might use some combination of protocol ID, port numbers(s), application layer protocol headers, IP address(es), VLAN ID(s) and even packet timing.

Not end-to-end (network-network): Very few networks honour a DSCP received from a neighbouring network. Typically a network will zero (bleach) the Diffserv field from all neighbouring networks at an interconnection point. Sometimes bilateral arrangements are made between networks, such that the receiving network remarks some DSCPs to those it uses for roughly equivalent services. The likelihood that a DSCP will be bleached or ignored depends on the type of DSCP:

Local-use DSCP: These tend to be used to implement application-specific network policies, but a bilateral arrangement to remark certain DSCPs is often applied to DSCPs in the local-use range simply because it is easier not to change all of a network's internal configurations when a new arrangement is made with a neighbour;

Global-use DSCP: These do not tend to be honoured across network interconnections more than local-use DSCPs. However, if two networks decide to honour certain of each other's DSCPs, the reconfiguration is a little easier if both of their globally recognised services are already represented by the relevant global-use DSCPs.

Note that today a global-use DSCP gives little more assurance of end-to-end service than a local-use DSCP. In future the global-use range might give more assurance of end-to-end service than local-use, but it is unlikely that either assurance will be high, particularly given the hosts are included in the end-to-end path.

Not all tunnels: Diffserv codepoints are often not propagated to the outer header when a packet is encapsulated by a tunnel header. DSCPs are propagated to the outer of uniform mode tunnels, but not pipe mode [RFC2983], and pipe mode is fairly common.

ECN hard in some lower layers:: Because this approach uses both the Diffserv and ECN fields, an AQM wil only work at a lower layer if both can be supported. If individual network operators wished to deploy an AQM at a lower layer, they would usually propagate an IP Diffserv codepoint to the lower layer, using for example IEEE 802.1p. However, the ECN capability is harder to propagate down to lower layers because few lower layers support it.

Pros:

Could migrate to e2e: If all usage of classic ECN migrates to usage of L4S, the DSCP would become redundant, and the ECN capability alone could eventually identify L4S packets without the interconnection problems of Diffserv detailed below, and without having permanently consumed more than one codepoint in the IP header. Although the DSCP does not generally function as an end-to-end identifier (see below), it could be used initially by individual ISPs to introduce the L4S service for their own locally generated traffic;

A.3. ECN capability alone

Definition:

This approach uses ECN capability alone as the L4S identifier. It is only feasible if classic ECN is not widely deployed. The specific definition of codepoints would be:

* Any ECN codepoint other than Not-ECT would signify an L4S-capable sender, which in turn would indicate that both transports supported accurate ECN feedback [I-D.ietf-tcpm-accecn-reqs];

* ECN codepoints would not be used for classic ECN, and the classic network service would only be used for Not-ECT packets.

This approach would only be feasible if

A. it was generally agreed that there was little chance of any classic ECN deployment in any network;

B. developers of operating systems for user devices would only enable ECN by default once the TCP stack implemented accurate ECN [I-D.ietf-tcpm-accecn-reqs] including requesting it by default;

C. hosts would only negotiate accurate ECN if they supported L4S behaviour. In other words, developers of client OSs would all have to agree not to encourage further deployment of classic ECN.

Cons:

Near-infeasible deployment constraints: The constraints for deployment above represent a highly unlikely set of circumstances, but not completely impossible. If, despite the above measures, a pair of hosts did negotiate to use classic ECN, their packets would be classified into the same queue as L4S traffic, and if they had to compete with a long-running L4S flow they would get a very small capacity share;

ECN hard in some lower layers: See the same issue with "ECT(1) and CE codepoints" (Appendix A.1);

Non-L4S service for control packets: See the same issue with "ECT(1) and CE codepoints" (Appendix A.1).

Pros:

Consumes no additional codepoints: The ECT(1) codepoint and all spare Diffserv codepoints would remain available for future use;

Should work e2e: As with "ECT(1) and CE codepoints" (Appendix A.1);

Should work in tunnels: As with "ECT(1) and CE codepoints" (Appendix A.1).

A.4. Protocol ID

It has been suggested that a new ID in the IPv4 Protocol field or the IPv6 Next Header field could identify L4S packets. However this approach is ruled out by numerous problems:

- o A new protocol ID would need to be paired with the old one for each transport (TCP, SCTP, UDP, etc.);
- o In IPv6, there can be a sequence of Next Header fields, and it would not be obvious which one would be expected to identify a network service like L4S;
- o A new protocol ID would rarely provide an end-to-end service, because It is well-known that new protocol IDs are often blocked by numerous types of middlebox;
- o The approach is not a solution for AQMs below the IP layer;

A.5. Source or destination addressing

Locally, a network operator could arrange for L4S service to be applied based on source or destination addressing, e.g. packets from its own data centre and/or CDN hosts, packets to its business customers, etc. It could use addressing at any layer, e.g. IP addresses, MAC addresses, VLAN IDs, etc. Although addressing might be a useful tactical approach for a single ISP, it would not be a feasible approach to identify an end-to-end service like L4S. Even for a single ISP, it would require packet classifiers in buffers to be dependent on changing topology and address allocation decisions elsewhere in the network. Therefore this approach is not a feasible solution.

A.6. Summary: Merits of Alternative Identifiers

Table 1 provides a very high level summary of the pros and cons detailed against the schemes described respectively in Appendix A.2, Appendix A.3 and Appendix A.1, for six issues that set them apart.

Issue	DSCP + ECN		ECN	ECT(1) + CE	
	initial	eventual	initial	initial	eventual
end-to-end	N . .	. ? .	. . Y	. . Y	. . Y
tunnels	. O .	. O .	. . ?	. . ?	. . Y
lower layers	N . .	. ? .	. O .	. O .	. . ?
codepoints	N . .	. . ?	. . Y	N . .	. . ?
reordering	. . Y	. . Y	. . Y	. O .	. . ?
ctrl pkts	. . Y	. . Y	. O .	. O .	. . ?
			Note 1		

Note 1: Only feasible if classic ECN is obsolete.

Table 1: Comparison of the Merits of Three Alternative Identifiers

The schemes are scored based on both their capabilities now ('initial') and in the long term ('eventual'). The 'ECN' scheme shares the 'eventual' scores of the 'ECT(0) + CE' scheme. The scores are one of 'N, O, Y', meaning 'Poor', 'Ordinary', 'Good' respectively. The same scores are aligned vertically to aid the eye. A score of "?" in one of the positions means that this approach might optimisitically become this good, given sufficient effort. The table is not meant to be understandable without referring to the text.

Appendix B. Potential Competing Uses for the ECT(1) Codepoint

The ECT(1) codepoint of the ECN field has already been assigned once for experimental use [RFC3540]. ECN is probably the only remaining field in the Internet Protocol that is common to IPv4 and IPv6 and still has potential to work end-to-end, with tunnels and with lower layers. Therefore, ECT(1) should not be reassigned to a different experimental use without carefully assessing competing potential uses. These fall into the following categories:

B.1. Integrity of Congestion Feedback

Receiving hosts can fool a sender into downloading faster by suppressing feedback of ECN marks (or loss if retransmissions are not necessary or available otherwise). [RFC3540] proposes that a TCP sender could set either ECT(0) or ECT(1) in each packet of a flow and remember the pattern, termed the ECN nonce. If any packet is lost or congestion marked, the receiver will miss that bit of the sequence. An ECN Nonce receiver has to feed back the least significant bit of

the sum, so it cannot suppress feedback of a loss or mark without a 50-50 chance of guessing the sum incorrectly.

As far as is known, the ECN Nonce has never been deployed, and it was only implemented for a couple of testbed evaluations. It would be nearly impossible to deploy now, because any misbehaving receiver can simply opt-out, which would be unremarkable given all receivers currently opt-out.

Other ways to protect TCP feedback integrity have since been developed that do not consume any extra codepoints. For instance:

- o the sender can test the integrity of the receiver's feedback by occasionally setting the IP-ECN field to a value normally only set by the network. Then it can test whether the receiver's feedback faithfully reports what it expects [I-D.moncaster-tcpm-rcv-cheat]. This works for loss and it will work for the accurate ECN feedback [I-D.ietf-tcpm-accecn-reqs] intended for L4S;

- o A network can enforce a congestion response to its ECN markings (or packet losses) by auditing congestion exposure (ConEx) [I-D.ietf-conex-abstract-mech]. Whether the receiver or a downstream network is suppressing congestion feedback or the sender is unresponsive to the feedback, or both, ConEx audit can neutralise any advantage that any of these three parties would otherwise gain.

ECN in RTP [RFC6679] is defined so that the receiver can ask the sender to send all ECT(0); all ECT(1); or both randomly. It recommends that the receiver asks for ECT(0), which is the default. The sender can choose to ignore the receiver's request. A rather complex but optional nonce mechaism was included in early drafts of RFC 6679, but it was replaced with a statement that a nonce mechanism is not specified, explaining that misbehaving receivers could opt-out anyway. RFC 6679 as published gives no rationale for why ECT(1) or 'random' might be needed, but it warns that 'random' would make header compression highly inefficient. The possibility of using ECT(1) may have been left in the RFC to allow a nonce mechanism to be added later.

Therefore, it seems unlikely that anyone has implemented the optional use of ECT(1) for RTP, it even if they have, it seems even less likely that any deployment actually uses it. However these assumptions will need to be verified.

B.2. Notification of Less Severe Congestion than CE

Various researchers have proposed to use ECT(1) as a less severe congestion notification than CE, particularly to enable flows to fill available capacity more quickly after an idle period, when another flow departs or when a flow starts, e.g. VCP [VCP], Queue View (QV) [QV] {ToDo: Jonathan Morton's ELR if relevant once the promised write-up appears}.

Before assigning ECT(1) as an identifer for L4S, we must carefully consider whether it might be better to hold ECT(1) in reserve for future standardisation of rapid flow acceleration, which is an important and enduring problem [RFC6077].

Pre-Congestion Notification (PCN) is another scheme that assigns alternative semantics to the ECN field. It uses ECT(1) to signify a less severe level of pre-congestion notification than CE [RFC6660]. However, the ECN field only takes on the PCN semantics if packets carry a Diffserv codepoint defined to indicate PCN marking within a controlled environment. PCN is required to be applied solely to the outer header of a tunnel across the controlled region in order not to interfere with any end-to-end use of the ECN field. Therefore a PCN region on the path would not interfere with any of the L4S service identifiers proposed in Appendix A.

Authors' Addresses

Koen De Schepper
Bell Labs
Antwerp
Belgium

Email: koen.de_schepper@alcatel-lucent.com
URI: https://www.bell-labs.com/usr/koen.de_schepper

Bob Briscoe (editor)
Simula Research Lab

Email: ietf@bobbriscoe.net
URI: http://bobbriscoe.net/

Ing-jyh Tsang
Bell Labs
Antwerp
Belgium

Email: ing-jyh.tsang@alcatel-lucent.com

INTERNET-DRAFT
Intended Status: Informational
Expires: April 17, 2016

T. Herbert
Facebook
L. Yong
Huawei USA

October 15, 2015

UDP Magic Numbers
draft-herbert-udp-magic-numbers-01

Abstract

This specification defines magic numbers in UDP which allow a node to determine or confirm the protocol contained in a UDP payload. This is primarily applicable for encapsulation and transport protocols encapsulated within UDP where intermediate devices, such as middle boxes, need to parse these protocols for providing service. Magic numbers can also be used to multiplex different UDP encapsulated protocols over the same UDP port.

Status of this Memo

Copyright and License Notice

Table of Contents

1 Introduction

Several transport and encapsulation protocols have been defined to be encapsulated within UDP [RFC0768]. In this model, the payload of a UDP packet contains a protocol header and payload for an encapsulated protocol. Transport protocols encapsulated in UDP include QUIC [QUIC], SCTP-in-UDP [RFC6951], and SPUD [I-D.hildebrand-spud-prototype]. Encapsulation protocols include Geneve [I-D.ietf-nvo3-geneve], VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe], GUE [I-D.ietf-nvo3-gue], MPLS-in-UDP [RFC7510], and GRE-in-UDP [I-D.ietf-tsvwg-gre-in-udp-encap]. For various reasons, intermediate devices in a network may want to parse these protocols. For instance, a middlebox would need to parse an encapsulated transport protocol to implement a stateful firewall. To parse the encapsulated protocol in a UDP packet, a node must positively identify the encapsulated protocol.

The destination UDP port number is commonly used to interpret the contents of a UDP payload, however this is problematic in intermediate devices for several reasons:

- Port numbers can only be correctly interpreted by the endpoints. Interpretation by intermediate devices in the network may be incorrect ([RFC7605]]).

- Encapsulation and transport protocols will usually have assigned UDP ports, but they are not restricted to use only those.

- UDP encapsulated protocols may use a "substrate" protocol header as espoused in SPUD. Use of a substrate header may be common across several port numbers. Configuring each network device for each port that uses the substrate could be cumbersome.

This specification describes UDP magic numbers which allows network nodes to identify UDP encapsulated protocols without relying solely on UDP port numbers. A UDP magic number is a protocol specific, constant value which is logically inserted between the UDP header and the encapsulated protocol header. If a node matches the magic number in a packet to a known protocol’s magic number, then it can parse the encapsulated payload per the matched protocol. Each UDP encapsulated protocol uses a different magic number which allows multiplexing multiple encapsulated protocols over the same UDP port.

Note that the use of magic numbers is inherently probabilistic. It is possible that a UDP packet may have a payload that inadvertently matches a magic number. The magic number is defined to minimize the probability of this occurring (1/2^^64 assuming that UDP data has a random distribution), nevertheless the probability is non-zero. The consequences of incorrectly matching a UDP packet should be

considered for each UDP encapsulated protocol. An encapsulated protocol may include its own verification to ensure correct interpretation.

The use of magic numbers to identify UDP encapsulated protocols was specified in the SPUD prototype protocol ([I-D.hildebrand-spud-prototype]) and in "Session Traversal Utilities for NAT (STUN)" ([RFC5389]). This proposal generalizes the concept.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2 Magic number format

The UDP magic number is a sixty-four bit value that includes a fixed constant, an encapsulated protocol type, and a checksum. The magic number within a UDP packet is diagrammed below.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Source port            |      Destination port         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Length              |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Magic value = 0xffd871a2                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Xor'ed protocol       |       Xor'ed checksum         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                      Encapsulated protocol                    ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The fields of the magic number are:

- o Magic value: A fixed constant of 0xffd871a2. This value is the same for all encapsulated protocol types.

- o Xor'ed protocol: Indicates the protocol type of the encapsulated protocol. The value in the field is a protocol type number exclusive or'ed with 0x36b4.

- o Xor'ed Checksum: Indicates the standard one's complement

checksum over the magic number (including the Magic value and Xor'ed Protocol fields). The value in this field is the calculated checksum exclusive or'ed with 0x5ce9.

- o Encapsulated protocol: This contains the header and payload of the encapsulated protocol. The type for the protocol is indicated in the Xor'ed protocol field.

2.1 Magic value

The first byte of the Magic value field is 0xff and the other three bytes are a randomly chosen constant.

For UDP encapsulated protocols that allow magic number use to be optional, the magic number must be clearly distinguishable from a valid header. Each such protocol must declare that a header which would match the associated magic number is invalid. The value of 0xff as the first byte in the magic number was chosen as a likely value that would indicate an invalid header. It is common that the first byte of a protocol header contains a version number, and most protocols have not gotten past version zero. So if a magic number is received by a node that does not yet support magic numbers, the UDP payload would likely be interpreted as a protocol header with a bad version number; this should result in dropping the packet and not misinterpreting it. In this way, the use of magic numbers can be enabled for many existing protocols with forward compatibility.

2.2 Protocol types

Protocol types can generally refer to any encapsulation, transport, substrate, or application specific protocol that is encapsulated in UDP for which intermediate devices might need to parse. A protocol type number is encoded in UDP magic numbers to allow intermediate devices to distinguish different payload types while still using a common magic number format.

Protocol types are indicated by sixteen bits numbers, and the space is divided into three regions.

Numbers 0-49151 are reserved to mirror the assigned UDP port number space. If a port number is assigned to a UDP encapsulated protocol, that same number can be used as the protocol type number. This is allowed for convenience, there is no required correlation between protocol type numbers and UDP port numbers.

Numbers 49152-57343 are reserved for assigned protocol types.

Numbers 57344-65535 are reserved for private protocol types.

The Xor'ed protocol field in a magic number is a protocol type number exclusive or'ed with 0x36b4.

2.3 Magic number checksum

The magic number checksum is calculated as the standards one complement checksum computed over the sixty-four bit magic number where the Xor'ed checksum field is set to zero for the purposes of calculation. The checksum calculation covers the Magic value and the Xor'ed protocol fields. The Xor'ed checksum field is set to the result of the calculation exclusive or'ed with 0x5ce9.

Note that the magic number checksum is performed over constant fields and is itself a constant value per protocol type. An implementation should not need to perform this calculation when processing packets. Appendix A demonstrates how the checksum is applied to create a magic number constant for Generic UDP Encapsulation.

The magic number checksum may be used to validate the presence of a well formed UDP magic number. This is demonstrated in Appendix B.

3 Usage

This section describes the processing of UDP magic numbers on end hosts and intermediate devices.

3.1 End hosts

The use of UDP magic numbers is enabled on a per port basis. Magic numbers may be required for every UDP packet sent on a port, or may be optional. If a UDP port is assigned to a single protocol, the magic number in packets sent to that port is the one assigned to the protocol. If different encapsulated protocols are multiplexed on the same UDP port, magic numbers for those protocols will be used.

3.1.1 Required magic numbers

If magic numbers are required for a UDP port, a sender must set the magic number in any packets sent to the destination port. A receiver must check for a valid magic number. If the magic number is valid, that is the Magic value is correct and the protocol type is supported by the receiver for the port, then the packet is accepted. Otherwise, the magic number is not matched so the packet is dropped.

3.1.2 Optional magic numbers

When magic numbers are optional for a UDP port, a receiver must check if a magic number is present in a received packet. If a magic number

is matched for a protocol type supported by the receiver, then the packet must be accepted and the Encapsulated protocol in the packet is processed according to the protocol type. If the magic number is not matched, the packet is still accepted and the UDP payload is processed as a protocol type implied by the port number.

If it is not feasible in a protocol to distinguish a magic number from a valid header (MPLS-in-UDP for instance), UDP magic numbers cannot be optional on the protocol's port number. They can be used on a separate port number for which magic numbers would be required.

3.1.3 Use with DTLS

UDP magic numbers are intended to occupy the first bytes of the UDP payload to facilitate interpretation at middleboxes. When they are used with DTLS [RFC6347], the magic number must precede the DTLS headers. The protocol type in the magic number would refer to the payload type contained in DTLS.

3.2 Intermediate devices

Intermediate devices may match magic numbers in two ways:

- Match both the destination port and magic numbers associated with the port.

- Match magic numbers across a range (possibly all) of ports.

Matching both the port and magic number is recommended. This is feasible in cases where a UDP encapsulated protocol has an assigned port number. Matching the port number and magic number significantly reduces the possibility of misinterpreting a packet.

Matching just the magic number and not a port may be done when UDP encapsulated protocols are used on unassigned ports, or configuring port numbers on intermediate devices is prohibitive.

In either case, if a middlebox is able to match a magic number it may parse the encapsulated payload of the packet for the associated protocol.

If a middle box does not match a magic number for a packet it should follow default processing for UDP packets. If magic numbers are known to be required for a port, a middlebox may perform some alternative processing when the magic number is not present. This alternative processing should not be more restrictive than had the packet been sent to another arbitrary UDP port. In particular, if UDP packets for other ports would not be dropped, failure to match a magic number

should not result in the packet being dropped.

4 Security Considerations

UDP magic numbers are not a security mechanism and should not increase security risk.

5 IANA Considerations

IANA will be requested to create a "UDP Magic Number Protocol Type" registry to allocate protocol types. This shall be a registry of 16-bit values along with descriptive strings. The allocation ranges are described in section 2.2.

6 References

6.1 Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC1776] Crocker, S., "The Address is the Message", RFC 1776, DOI 10.17487/RFC1776, April 1 1995, <http://www.rfc-editor.org/info/rfc1776>.

[TRUTHS] Callon, R., "The Twelve Networking Truths", RFC 1925, DOI 10.17487/RFC1925, April 1 1996, <http://www.rfc-editor.org/info/rfc1925>.

6.2 Informative References

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013, <http://www.rfc-editor.org/info/rfc6951>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

[QUIC] Roskind, J., "QUIC: Multiplexed Stream Transport Over UDP", http://www.ietf.org/proceedings/88/slides/slides-88-

tsvarea-10.pdf

[I-D.hildebrand-spud-prototype] Hildebrand, J. and Trammell, B. "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in preogress), March 2015.

[I-D.ietf-nvo3-geneve] Gross, J., Sridhar, T., Garg, P., Wright, C., Ganga, I., Agarwal, P., Duda, K., Dutt, D., and J. Hudson, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-00, May 2015.

[I-D.ietf-nvo3-vxlan-gpe] Quinn, P., Manur, R., Kreeger, L., Lewis, D., Maino, F., Smith, M., Agarwal, P., Yong, L., Xu, X, Elzur, U., Garg, P., and Melman, D. "Generic Protocol Extension for VXLAN" draft-ietf-nvo3-vxlan-gpe-00draft-quinn-vxlan-gpe-00, February 2015

[I-D.ietf-nvo3-gue] Herbert, T., Yong, L., and Zia, O., "Generic UDP Encapsulation", draft-ietf-nvo3-gue-01 (work in progress), June 2015.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, April 2015, <http://www.rfc-editor.org/info/rfc7510>.

[I-D.ietf-tsvwg-gre-in-udp-encap] Crabbe, E., Yong, L., Xu, X., and Herbert, T. "GRE-in-UDP Encapsulation", draft-ietf-tsvwg-gre-in-udp-encap-07, July 2015

[RFC7605] Touch, J., "Recommendations on Using Assigned Transport Port Numbers", BCP 165, RFC 7605, DOI 10.17487/RFC7605, August 2015, <http://www.rfc-editor.org/info/rfc7605>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[EVILBIT] Bellovin, S., "The Security Flag in the IPv4 Header", RFC 3514, DOI 10.17487/RFC3514, April 1 2003, <http://www.rfc-editor.org/info/rfc3514>.

[RFC5513] Farrel, A., "IANA Considerations for Three Letter Acronyms", RFC 5513, DOI 10.17487/RFC5513, April 1 2009, <http://www.rfc-editor.org/info/rfc5513>.

[RFC5514] Vyncke, E., "IPv6 over Social Networks", RFC 5514, DOI 10.17487/RFC5514, April 1 2009, <http://www.rfc-

editor.org/info/rfc5514>.

Appendix A: Example of creating a UDP magic number

This section demonstrates how a magic can be created for a UDP encapsulated protocol. For this example we consider Generic UDP Encapsulation (GUE), and assume that the assigned port number is used as the protocol type number.

The assigned port number for GUE in 6080 or 0x17c0 in hexadecimal. So the value of the Xor'ed protocol field is:

```
0x17c0 ^ 0x36b4 = 0x2174
```

To compute the magic checksum we first sum the words of the Magic value and the Xor'ed protocol field value computed above:

```
0xffd8 + 0x71a2 + 0x2174 = 0x192ee
```

The result is folded and then complemented:

```
(0x92ee + 1) ^ 0xffff = 0x6d10
```

So the value in the Xor'ed checksum field is:

```
0x6d10 ^ 0x5ce9 = 0x31f9
```

Thus the full 64 bit magic number value for GUE is:

```
0xffd871a2:0x217431f9
```

Appendix B: Checking magic numbers

This section provides some guidelines for how to check magic numbers.

B.1 Matching a single magic number

When a port supports precisely one protocol type there is only one magic number to check. This will be a common case at a receiver where magic numbers are enabled for encapsulated protocols that have assigned ports. Receiver processing in pseudo code may be:

```
dataptr = UDP_payload_ptr
good_magic = false
PROTO_MAGIC_NUMBER = Pre computed 64 bit value for protocol type

if (UDP_payload_length >= 8 &&
    memcmp(UDP_payload_ptr, PROTO_MAGIC_NUMBER, 8) == 0) {
```

```
              /* Magic number matched, skip it for further processing */
              dataptr += 8
              good_magic = true
      }

      if (good_magic || magic_numbers_are_optional)
              process_packet(dataptr)
      else
          /* Handle bad packet */
```

B.2 Matching against a set of magic numbers

A host needs to check against a set of magic numbers when different encapsulated protocols are multiplexed over a single port, and an intermediate device checks against a set when matching magic numbers across a range of ports. In either case, the typical method is to check the first four bytes of the UDP payload against the constant magic number value. If this is a match then the protocol type number is extracted and a lookup is performed to find a context. If a context is found, the checksum field in the packet is compared against a precomputed value in the context. In pseudo code this is:

```
      dataptr = UDP_payload_ptr;
      good_magic = false;

      if (UDP_payload_length >= 8 &&
          *(u32 *)UDP_payload_ptr == 0xffd871a2) {
             proto = *(u16 *)(UDP_payload_ptr + 4) ^ 0x36b4
             checksum = *(u16 *)(UDP_payload_ptr + 6)
             ctx = protocol_lookup(proto)
             if (ctx && checksum == ctx->checksum) {
                    /* Protocol found and matched */
                    good_magic = true
                    dataptr += 8;
             }
      }

      if (good_magic)
             process_as_protocol(dataptr, proto);
      else
             /* Handle bad packet */
```

B.3 Magic number validation

A node can validate that a magic number is well formed for any protocol. This requires checking the Magic value is correct and verifying the checksum. In pseudo code this would be:

```
good_magic = false
u16 checksum(start, len) /* Checksum function */
if (UDP_payload_length >= 8 &&
    *(u32 *)UDP_payload_ptr == 0xffd871a2) {
        csum = checksum(UDP_payload_ptr, 6)
        if (csum ^ 0x5ce9 == *(u16 *)(UDP_payload_ptr + 6))
               good_magic = true
}
```

Authors' Addresses

Tom Herbert
Facebook
1 Hacker Way
Menlo Park, CA
US

EMail: tom@herbertland.com

Lucy Yong
Huawei USA
5340 Legacy Dr.
Plano, TX 75024
US

Email: lucy.yong@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 27, 2016

G. Fairhurst
University of Aberdeen
M. Welzl
University of Oslo

November 24, 2015

The Benefits of using Explicit Congestion Notification (ECN)
draft-ietf-aqm-ecn-benefits-08

Abstract

The goal of this document is to describe the potential benefits when applications use a transport that enables Explicit Congestion Notification (ECN). The document outlines the principal gains in terms of increased throughput, reduced delay and other benefits when ECN is used over a network path that includes equipment that supports Congestion Experienced (CE) marking. It also discusses challenges for successful deployment of ECN. It does not propose new algorithms to use ECN, nor does it describe the details of implementation of ECN in endpoint devices (Internet hosts), routers or other network devices.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Internet Transports (such as TCP and SCTP) are implemented in endpoints (Internet hosts) and are designed to detect and react to network congestion. Congestion may be detected by loss of an IP packet or, if Explicit Congestion Notification (ECN) [RFC3168] is enabled, by the reception of a packet with a Congestion Experienced (CE) marking in the IP header. Both of these are treated by transports as indications of congestion. ECN may also be enabled by

other transports: UDP applications that provide congestion control may enable ECN when they are able to correctly process the ECN signals [ID.RFC5405.bis] (e.g., ECN with RTP [RFC6679]).

Active Queue Management (AQM) [RFC7567] is a class of techniques that can be used by network devices (a router, middlebox, or other device that forwards packets through the network) to manage the size of queues in network buffers.

A network device that does not support AQM typically uses a drop-tail policy to drop excess IP packets when its queue becomes full. The discard of packets is treated by transport protocols as a signal that indicates congestion on the end-to-end network path. End-to-end transports, such as TCP, can cause a low level of loss while seeking to share capacity with other flows. Although losses are not always due to congestion (loss may be due to link corruption, receiver-overrun, etc) end points have to conservatively presume that all loss is potentially due to congestion and reduce their rate. Observed loss therefore results in a congestion control reaction by the transport to reduce the maximum rate permitted by the sending endpoint.

ECN makes it possible for the network to signal the presence of incipient congestion without incurring packet loss, it lets the network deliver some packets to an application that would otherwise have been dropped if the application or transport did not support ECN. This packet loss reduction is the most obvious benefit of ECN, but it is often relatively modest. However, enabling ECN can also result in a number of beneficial side-effects, some of which may be much more significant than the immediate packet loss reduction from receiving CE-marking instead of dropping packets. Several benefits reduce latency (e.g., reduced Head-of-Line Blocking).

The use of ECN is indicated in the ECN field [RFC3168], carried in the packet header of all IPv4 and IPv6 packets. This field may be set to one of four values shown in Table 1. The not-ECT codepoint ’00’ indicates a packet that is not using ECN. The ECT(0) codepoint ’01’ and the ECT(1) codepoint ’10’ both indicate that the transport protocol using the IP layer supports the use of ECN. The CE codepoint ’11’ is set by an ECN-capable network device to indicate congestion to the transport endpoint.

ECN FIELD		Name
0	0	Not-ECT
0	1	ECT(1)
1	0	ECT(0)
1	1	CE

Table 1: The ECN Field in the IP Packet Header (based on [RFC3168]).

When an application uses a transport that enables use of ECN [RFC3168], the transport layer sets the ECT(0) or ECT(1) codepoint in the IP header of packets that it sends. This indicates to network devices that they may mark, rather than drop the ECN-capable IP packets. An ECN-capable network device can then signal incipient congestion (network queueing) at a point before a transport experiences congestion loss or high queuing delay. The marking is generally performed as the result of various AQM algorithms [RFC7567], where the exact combination of AQM/ECN algorithms does not need to be known by the transport endpoints.

The focus of the document is on usage of ECN by transport and application layer flows, not its implementation in endpoint hosts, or in routers and other network devices.

1.1. Terminology

The following terms are used:

AQM: Active Queue Management.

CE: Congestion Experienced, a codepoint value ’11’ marked in the ECN field of the IP packet header.

ECN-capable IP Packet : A packet where the ECN field is set to a non-zero ECN value (i.e., with a ECT(0), ECT(1), or the CE codepoint).

ECN-capable network device : An ECN-capable network device may forward, drop, or queue an ECN-capable packet and may choose to CE-mark this packet when there is incipient congestion.

ECN-capable transport/application : A transport that sends ECN-capable IP Packets, and monitors reception of the ECN field and generates appropriate feedback to control the rate of the sending endpoint.

to provide end-to-end congestion control.

ECN field: A 2-bit field specified for use explicit congestion signalling in the IPv4 and IPv6 packet headers.

Endpoint: An Internet host that terminates a transport protocol connection across an Internet path.

Incipient Congestion: The detection of congestion when it is starting, perhaps by a network device noting that the arrival rate exceeds the forwarding rate.

Network device: A router, middlebox, or other device that forwards IP packets through the network.

non-ECN-capable: A network device or endpoint that does not interpret the ECN field. Such a device is not permitted to change the ECN codepoint.

not-ECN-capable IP Packet: An IP packet with the ECN field set to a value of zero ('00'). A not-ECN-capable packet may be forwarded, dropped or queued by a network device.

2. Benefit of using ECN to avoid Congestion Loss

An ECN-capable network device is expected to CE-mark an ECN-capable IP packet when an AQM method detects incipient congestion, rather than to drop the packet [RFC7567]. An application can benefit from this marking in several ways:

2.1. Improved Throughput

ECN seeks to avoid the inefficiency of dropping data that has already made it across at least part of the network path.

ECN can improve the throughput of an application, although this increase in throughput is often not the most significant gain. When an application uses a light to moderately loaded network path, the number of packets that are dropped due to congestion is small. Using an example from Table 1 of [RFC3649], for a standard TCP sender with a Round Trip Time, RTT, of 0.1 seconds, a packet size of 1500 bytes and an average throughput of 1 Mbps, the average packet drop ratio would be 0.02 (i.e., 1 in 50 packets). This translates into an approximate 2% throughput gain if ECN is enabled. (Note that in heavy congestion, packet loss may be unavoidable with, or without, ECN.)

2.2. Reduced Head-of-Line Blocking

Many Internet transports provide in-order delivery of received data segments to the applications they support. For these applications, use of ECN can reduce the delay that can result when these applications experience packet loss.

Packet loss may occur for various reasons. One cause arises when an AQM scheme drops a packet as a signal of incipient congestion. Whatever the cause of loss, a missing packet needs to trigger a congestion control response. A reliable transport also triggers retransmission to recover the lost data. For a transport providing in-order delivery, this requires that the transport receiver stalls (or waits) for all data that was sent ahead of a lost segment to be correctly received before it can forward any later data to the application. A loss therefore creates a delay of at least one RTT after a loss event before data can be delivered to an application. We call this Head-of-Line (HOL) blocking. This is the usual requirement for TCP and SCTP. (PR-SCTP [RFC3758], UDP [RFC0768][ID.RFC5405.bis], and DCCP [RFC4340] provide a transport that does not provide re-ordering).

By enabling ECN, a transport continues to receive in-order data when there is incipient congestion, and can pass this data to the receiving application. Use of ECN avoids the additional reordering delay in a reliable transport. The sender still needs to make an appropriate congestion-response to reduce the maximum transmission rate for future traffic, which usually will require a reduction in the sending rate [ID.RFC5405.bis].)

2.3. Reduced Probability of RTO Expiry

Some patterns of packet loss can result in a Retransmission Time Out (RTO), which causes a sudden and significant change in the allowed rate at which a transport/application can forward packets. Because ECN provides an alternative to drop for network devices to signal incipient congestion, this can reduce the probability of loss and hence reduce the likelihood of RTO expiry.

Internet transports/applications generally use a RTO timer as a last resort to detect and recover loss [ID.RFC5405.bis] [RFC5681]). Specifically, a RTO timer detects loss of a packet that is not followed by other packets, such as at the end of a burst of data segments or when an application becomes idle (either because the application has no further data to send or the network prevents sending further data, e.g., flow or congestion control at the transport layer). This loss of the last segment (or last few segments) of a traffic burst is also known as a "tail loss".

Standard transport recovery methods, such as Fast Recovery ([RFC5681], are often unable to recover from a tail loss. This is because the endpoint receiver is unaware that the lost segments were actually sent, and therefore generates no feedback [Fla13]. Retransmission of these segments therefore relies on expiry of a transport retransmission timer. This timer is also used to detect a lack of forwarding along a path. Expiry of the RTO therefore results in the consequent loss of state about the network path being used. This typically includes resetting path estimates such as the RTT, re-initialising the congestion window, and possibly updates to other transport state. This can reduce the performance of the transport until it again adapts to the path.

An ECN-capable network device cannot eliminate the possibility of tail loss, because a drop may occur due to a traffic burst exceeding the instantaneous available capacity of a network buffer or as a result of the AQM algorithm (overload protection mechanisms, etc [RFC7567]). However, an ECN-capable network device that observes incipient congestion may be expected to buffer the IP packets of an ECN-capable flow and set a CE-mark in one or more packet(s), rather than triggering packet drop. Setting a CE-mark signals incipient congestion without forcing the transport/application to enter retransmission timeout. This reduces application-level latency and can improve the throughput for applications that send intermittent bursts of data.

The benefit of avoiding retransmission loss is expected to be significant when ECN is used on TCP SYN/ACK packets [RFC5562] where the RTO interval may be large because TCP cannot base the timeout period on prior RTT measurements from the same connection.

2.4. Applications that do not Retransmit Lost Packets

A transport that enables ECN can receive timely congestion signals without the need to retransmit packets each time it receives a congestion signal.

Some latency-critical applications do not retransmit lost packets, yet may be able to adjust their sending rate following detection of incipient congestion. Examples of such applications include UDP-based services that carry Voice over IP (VoIP), interactive video, or real-time data. The performance of many such applications degrades rapidly with increasing packet loss and the transport/application may therefore employ mechanisms (e.g., packet forward error correction, data duplication, or media codec error concealment) to mitigate the immediate effect of congestion loss on the application. Some mechanisms consume additional network capacity, some require additional processing and some contribute additional path latency

when congestion is experienced. By decoupling congestion control from loss, ECN can allow transports that support these applications to reduce their rate before the application experiences loss from congestion. This can reduce the negative impact of triggering loss-hiding mechanisms with a direct positive impact on the quality experienced by the users of these applications.

2.5. Making Incipient Congestion Visible

A characteristic of using ECN is that it exposes the presence of congestion on a network path to the transport and network layers allowing information to be collected about the presence of incipient congestion.

Recording the presence of CE-marked packets can provide information about the current congestion level experienced on a network path. A network flow that only experiences CE-marking and no loss implies that the sending endpoint is experiencing only congestion. A network flow may also experience loss (e.g., due to queue overflow, AQM methods that protect other flows, link corruption or loss in middleboxes). When a mixture of CE-marking and packet loss is experienced, transports and measurements need to assume there is congestion [RFC7567]. An absence of CE-marks therefore does not indicate a path has not experienced congestion.

The reception of CE-marked packets can be used to monitor the level of congestion by a transport/application or a network operator. For example, ECN measurements are used by Congestion Exposure (ConEx) [RFC6789]. In contrast, metering packet loss is harder.

2.6. Opportunities for new Transport Mechanisms

ECN can enable design and deployment of new algorithms in network devices and Internet transports. Internet transports need to regard both loss and CE-marking as an indication of congestion. However, while the amount of feedback provided by drop ought naturally to be minimized, this is not the case for ECN. In contrast, an ECN-Capable network device could provide richer (more frequent and fine-grained) indication of its congestion state to the transport.

For any ECN-capable transport, the receiving endpoint needs to provide feedback to the transport sender to indicate that CE-marks have been received.[RFC3168] provides one method that signals once each round trip time that CE-marked packets have been received.

A receiving endpoint may provide more detailed feedback to the congestion controller at the sender (e.g., describing the set of received ECN codepoints, or indicating each received CE-marked

packet). Precise feedback about the number of CE-marks encountered is supported by the Real Time Protocol (RTP) when used over UDP [RFC6679] and has been proposed for SCTP [ST14] and TCP [ID.Acc.ECN].

More detailed feedback is expected to enable evolution of transport protocols allowing the congestion control mechanism to make a more appropriate decision on how to react to congestion. Designers of transport protocols need to consider not only how network devices CE-mark packets, but also how the control loop in the application/ transport reacts to reception of these CE-marked packets.

Benefit has been noted when packets are CE-marked early using an instantaneous queue, and if the receiving endpoint provides feedback about the number of packet marks encountered, an improved sender behavior has been shown to be possible, e.g, Datacenter TCP (DCTCP) [AL10]. DCTCP is targeted at controlled environments such as a datacenter. This is work-in-progress and it is currently unknown whether or how such behaviour could be safely introduced into the Internet. Any update to an Internet transport protocol requires careful consideration of the robustness of the behaviour when working with endpoints or network devices that were not designed for the new congestion reaction.

3. Network Support for ECN

For an application to use ECN requires that the endpoints first enable ECN within the transport being used, but also for all network devices along the path to at least forward IP packets that set a non-zero ECN codepoint.

ECN can be deployed both in the general Internet and in controlled environments:

- o ECN can be incrementally deployed in the general Internet. The IETF has provided guidance on configuration and usage in [RFC7567].

- o ECN may be deployed within a controlled environment, for example within a data centre or within a well-managed private network. This use of ECN may be tuned to the specific use-case. An example is DCTCP [AL10] [ID.DCTCP].

Early experience of using ECN across the general Internet encountered a number of operational difficulties when the network path either failed to transfer ECN-capable packets or inappropriately changed the ECN codepoints [BA11]. A recent survey reported a growing support for network paths to pass ECN codepoints [TR15].

The remainder of this section identifies what is needed for network devices to effectively support ECN.

3.1. The ECN Field

The current IPv4 and IPv6 specifications assign usage of 2 bits in the IP header to carry the ECN codepoint. This 2-bit field was reserved in [RFC2474] and assigned in [RFC3168].

[RFC4774] discusses some of the issues in defining alternate semantics for the ECN field, and specifies requirements for a safe coexistence in an Internet that could include routers that do not understand the defined alternate semantics.

Some network devices were configured to use a routing hash that included the set of 8 bits forming the now deprecated Type of Service (ToS) field [RFC1349]. The present use of this field assigns 2 of these bits to carry the ECN field. This is incompatible with use in a routing hash, because it could lead to IP packets that carry a CE-mark being routed over a different path to those packets that carried an ECT mark. The resultant reordering would impact the performance of transport protocols (such as TCP or SCTP) and UDP-based applications that are senstive to reordering. A network device that conforms to this older specification needs to be updated to the current specifications [RFC2474] to support ECN. Configuraton of network devices must note that the ECN field may be updated by any ECN-capable network device along a path.

3.2. Forwarding ECN-Capable IP Packets

Not all network devices along a path need to be ECN-capable (i.e., perform CE-marking). However, all network devices need to be configured not to drop packets solely because the ECT(0) or ECT(1) codepoints are used.

Any network device that does not perform CE-marking of an ECN-capable packet can be expected to drop these packets under congestion. Applications that experience congestion at these network devices do not see any benefit from enabling ECN. However, they may see benefit if the congestion were to occur within a network device that did support ECN.

3.3. Enabling ECN in Network Devices

Network devices should use an AQM algorithm that CE-marks ECN-capable traffic when making decisions about the response to congestion [RFC7567]. An ECN method should set a CE-mark on ECN-capable packets in the presence of incipient congestion. A CE-marked packet will be

interpreted as an indication of incipient congestion by the transport endpoints.

There is opportunity to design an AQM method for an ECN-capable network device that differs from an AQM method designed to drop packets. [RFC7567] states that the network device should allow this behaviour to be configurable.

[RFC3168] describes a method in which a network device sets the CE-mark at the time that the network device would otherwise have dropped the packet. While it has often been assumed that network devices should CE-mark packets at the same level of congestion at which they would otherwise have dropped them, [RFC7567] recommends that network devices allow independent configuration of the settings for AQM dropping and ECN marking. Such separate configuration of the drop and mark policies is supported in some network devices.

3.4. Co-existance of ECN and non-ECN flows

Network devices need to be able to forward all IP flows and provide appropriate treatment for both ECN and non-ECN traffic.

The design considerations for an AQM scheme supporting ECN needs to consider the impact of queueing during incipient congestion. For example, a simple AQM scheme could choose to queue ECN-capable and non-ECN capable flows in the same queue with an ECN scheme that CE-mark packets during incipient congestion. The CE-marked packets that remain in the queue during congestion can continue to contribute to queueing delay. In contrast, non-ECN-capable packets would normally be dropped by an AQM scheme under incipient congestion. This difference in queueing is one motivation for consideration of more advanced AQM schemes, and may provide an incentive for enabling flow isolation using scheduling [RFC7567]. The IETF is defining methods to evaluate the suitability of AQM schemes for deployment in the general Internet [ID.AQM.eval].

3.5. Bleaching and Middlebox Requirements to deploy ECN

Network devices should not be configured to change the ECN codepoint in the packets that they forward, except to set the CE-codepoint to signal incipient congestion.

Cases have been noted where an endpoint sends a packet with a non-zero ECN mark, but the packet is received by the remote endpoint with a zero ECN codepoint [TR15]. This could be a result of a policy that erases or "bleaches" the ECN codepoint values at a network edge (resetting the codepoint to zero). Bleaching may occur for various

reasons (including normalising packets to hide which equipment supports ECN). This policy prevents use of ECN by applications.

When ECN-capable IP packets, marked as ECT(0) or ECT(1), are remarked to non-ECN-capable (i.e., the ECN field is set to zero codepoint), this could result in the packets being dropped by ECN-capable network devices further along the path. This eliminates the advantage of using of ECN.

A network device must not change a packet with a CE mark to a zero codepoint, if the network device decides not to forward the packet with the CE-mark, it has to instead drop the packet and not bleach the marking. This is because a CE-marked packet has already received ECN treatment in the network, and remarking it would then hide the congestion signal from the receiving endpoint. This eliminates the benefits of ECN. It can also slow down the response to congestion compared to using AQM, because the transport will only react if it later discovers congestion by some other mechanism.

Prior to RFC2474, a previous usage assigned the bits now forming the ECN field as a part of the now deprecated Type of Service (ToS) field [RFC1349]. A network device that conforms to this older specification was allowed to remark or erase the ECN codepoints, and such equipment needs to be updated to the current specifications to support ECN.

3.6. Tunneling ECN and the use of ECN by Lower Layer Networks

Some networks may use ECN internally or tunnel ECN (e.g., for traffic engineering or security). These methods need to ensure that the ECN-field of the tunnel packets is handled correctly at the ingress and egress of the tunnel. Guidance on the correct use of ECN is provided in [RFC6040].

Further guidance on the encapsulation and use of ECN by non-IP network devices is provided in [ID.ECN-Encap].

4. Using ECN across the Internet

A receiving endpoint needs to report the loss it experiences when it uses loss-based congestion control. So also, when ECN is enabled, a receiving endpoint must correctly report the presence of CE-marks by providing a mechanism to feed this congestion information back to the sending endpoint , [RFC3168], [ID.RFC5405.bis], enabling the sender to react to experienced congestion. This mechanism needs to be designed to operate robustly across a wide range of Internet path characteristics. This section describes partial deployment, how ECN-enabled endpoints can continue to work effectively over a path that

experiences misbehaving network devices or when an endpoint does not correctly provide feedback of ECN congestion information.

4.1. Partial Deployment

Use of ECN is negotiated between the endpoints prior to using the mechanism.

ECN has been designed to allow incremental partial deployment [RFC3168]. Any network device can choose to use either ECN or some other loss-based policy to manage its traffic. Similarly, transport/application negotiation allows senders and receiving endpoints to choose whether ECN will be used to manage congestion for a particular network flow.

4.2. Detecting whether a Path Really Supports ECN

Internet transport and applications need to be robust to the variety and sometimes varying path characteristics that are encountered in the general Internet. They need to monitor correct forwarding of ECN over the entire path and duration of a session.

To be robust, applications and transports need to be designed with the expectation of heterogeneous forwarding (e.g., where some IP packets are CE-marked by one network device, and some by another, possibly using a different AQM algorithm, or when a combination of CE-marking and loss-based congestion indications are used. ([ID.AQM.eval] describes methodologies for evaluating AQM schemes.)

A transport/application also needs to be robust to path changes. A change in the set of network devices along a path could impact the ability to effectively signal or use ECN across the path, e.g., when a path changes to use a middlebox that bleaches ECN codepoints (see Section 3.5).

A sending endpoint can check that any CE-marks applied to packets received over the path are indeed delivered to the remote receiving endpoint and that appropriate feedback is provided. (This could be done by a sender setting known a CE codepoint for specific packets in a network flow and then checking whether the remote endpoint correctly reports these marks [ID.Fallback], [TR15].) If a sender detects persistent misuse of ECN, it needs to fall back to using loss-based recovery and congestion control. Guidance on a suitable tranport reaction is provided in [ID.Fallback].

4.3. Detecting ECN Receiver Feedback Cheating

Appropriate feedback requires that the endpoint receiver does not try to conceal reception of CE-marked packets in the ECN feedback information provided to the sending endpoint [RFC7567]. Designers of applications/transports are therefore encouraged to include mechanisms that can detect this misbehavior. If a sending endpoint detects that a receiver is not correctly providing this feedback, it needs to fall back to using loss-based recovery instead of ECN.

5. Summary: Enabling ECN in Network Devices and Hosts

This section summarises the benefits of deploying and using ECN within the Internet. It also provides a list of prerequisites to achieve ECN deployment.

Application developers should where possible use transports that enable ECN. Applications that directly use UDP need to provide support to implement the functions required for ECN [ID.RFC5405.bis]. Once enabled, an application that uses a transport that supports ECN will experience the benefits of ECN as network deployment starts to enable ECN. The application does not need to be rewritten to gain these benefits. Table 2 summarises the key benefits.

Section	Benefit
2.1	Improved throughput
2.2	Reduced Head-of-Line blocking
2.3	Reduced probability of RTO Expiry
2.4	Applications that do not retransmit lost packets
2.5	Making incipient congestion visible
2.6	Opportunities for new transport mechanisms

Table 2: Summary of Key Benefits

Network operators and people configuring network devices should enable ECN [RFC7567].

Prerequisites for network devices (including IP routers) to enable use of ECN include:

- o A network device that updates the ECN field in IP packets must use IETF-specified methods (see Section 3.1).

- o A network device may support alternate ECN semantics (see Section 3.1).

- o A network device must not choose a different network path solely because a packet carries has a CE-codepoint set in the ECN Field, CE-marked packets need to follow the same path as packets with an ECT(0) or ECT(1) codepoint (see Section 3.1).Network devices need to be configured not to drop packets solely because the ECT(0) or ECT(1) codepoints are used (see Section 3.2).

- o A network device must not change a packet with a CE mark to a not-ECN-capable codepoint ('00'), if the network device decides not to forward the packet with the CE-mark, it has to instead drop the packet and not bleach the marking (see Section 3.5).

- o An ECN-capable network device should correctly update the ECN codepoint of ECN-capable packets in the presence of incipient congestion (see Section 3.3).

- o Network devices need to be able to forward both ECN-capable and not-ECN-capable flows (see Section 3.4).

Prerequisites for network endpoints to enable use of ECN include:

- o An application should use an Internet transport that can set and receive ECN marks (see Section 4).

- o An ECN-capable transport/application must return feedback indicating congestion to the sending endpoint and perform an appropriate congestion response (see Section 4).

- o An ECN-capable transport/application should detect paths where there is there is persistent misuse of ECN and fall back to not sending ECT(0) or ECT(1) (see Section 4.2).

- o Designers of applications/transports are encouraged to include mechanisms that can detect and react appropriately to misbehaving receivers that fail to report CE-marked packets (see Section 4.3).

6. Acknowledgements

The authors were part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed are solely those of the authors.

The authors would like to thank the following people for their comments on prior versions of this document: Bob Briscoe, David

Collier-Brown, Colin Perkins, Richard Scheffenegger, Dave Taht, Wes Eddy, Fred Baker, Mikael Abrahamsson, Mirja Kuehlewind, John Leslie, and other members of the TSVWG and AQM working groups.

7. IANA Considerations

XX RFC Ed - PLEASE REMOVE THIS SECTION XXX

This memo includes no request to IANA.

8. Security Considerations

This document introduces no new security considerations. Each RFC listed in this document discusses the security considerations of the specification it contains.

9. Revision Information

XXX RFC-Ed please remove this section prior to publication.

Revision 00 was the first WG draft.

Revision 01 includes updates to complete all the sections and a rewrite to improve readability. Added section 2. Author list reversed, since Gorry has become the lead author. Corrections following feedback from Wes Eddy upon review of an interim version of this draft.

Note: Wes Eddy raised a question about whether discussion of the ECN Pitfalls could be improved or restructured - this is expected to be addressed in the next revision.

Revision 02 updates the title, and also the description of mechanisms that help with partial ECN support.

We think this draft is ready for wider review. Comments are welcome to the authors or via the IETF AQM or TSVWG mailing lists.

Revision 03 includes updates from the mailing list and WG discussions at the Dallas IETF meeting.

The section "Avoiding Capacity Overshoot" was removed, since this refers primarily to an AQM benefit, and the additional benefits of ECN are already stated. Separated normative and informative references

Revision 04 (WG Review during WGLC)

Updated the abstract.

Added a table of contents.

Addressed various (some conflicting) comments during WGLC with new text.

The section on Network Support for ECN was moved, and some suggestions for rewording sections were implemented.

Decided not to remove section headers for 2.1 and 2.2 - to ensure the document clearly calls-out the benefits.

Updated references. Updated text to improve consistency of terms and added definitions for key terms.

Note: The group suggested this document should not define recommendations for end hosts or routers, but simply state the things needs to enable deployment to be successful.

Revision 05 (after WGLC comments)

Updated abstract to avoid suggesting that this describes new methods for deployment.

Added ECN-field definition, and sorted terms in order.

Added an opening para to each "benefit" to say what this is. Sought to remove redundancy between sections.

Added new section on Codepoints to avoid saying the same thing twice.

Reworked sections 3 and 4 to clarify discussion and to remove unnecessary text.

Reformatted Summary to refer to sections describing things, rather than appear as a list of new recommendations. Reordered to match the new document order.

Note: This version expects an update to RFC5405.bis that will indicate UDP ECN requirements (normative).

Revision 06

Corrections from Miria.

Revision 07

Update to include IESG feedback from: Spencer, Dan, Benoit, Joel. Corrected Non-ECN to Not-ECN where appropriate, added table of codepoints, clarified sentences describing "conservative" behaviour, added requirement to not do ToS-based routing (Junos enhanced hash), etc. Ammended Acknowledgments section.

Revision 08

Typo and definition correction from Bob Briscoe.

10. References

10.1. Normative References

[ID.RFC5405.bis] Eggert, Lars., Fairhurst, Gorry., and Greg. Shepherd, "Unicast UDP Usage Guidelines", 2015.

[RFC2474] "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers".

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <http://www.rfc-editor.org/info/rfc6040>.

[RFC7567] Baker, F. and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management", Internet-draft draft-ietf-aqm-recommendation-06, October 2014.

10.2. Informative References

[AL10] Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and M. Sridharan, "Data Center TCP (DCTCP)", SIGCOMM 2010, August 2010.

[BA11] Bauer, Steven., Beverly, Robert., and Arthur. Berger, "Measuring the State of ECN Readiness in Servers, Clients, and Routers, ACM IMC", 2011.

[Fla13] Flach, Tobias., Dukkipati, Nandita., Terzis, Andreas., Raghavan, Barath., Cardwell, Neal., Cheng, Yuchung., Jain, Ankur., Hao, Shuai., Katz-Bassett, Ethan., and Ramesh. Govindan, "Reducing web latency: the virtue of gentle aggression.", SIGCOMM 2013, October 2013.

[ID.Acc.ECN] Briscoe, Bob., Scheffeneger, Richard., and Mirja. Kuehlewind, "More Accurate ECN Feedback in TCP, Work-in-Progress".

[ID.AQM.eval] Kuhn, Nicolas., Natarajan, Preethi., Ros, David., and Naeem. Khademi, "AQM Characterization Guidelines (Work-in-progress, draft-ietf-aqm-eval-guidelines)", 2015.

[ID.DCTCP] Bensley, S., Eggert, Lars., and D. Thaler, "Microsoft’s Datacenter TCP (DCTCP): TCP Congestion Control for Datacenters (Work-in-progress, draft-bensley-tcpm-dctcp)", 2015.

[ID.ECN-Encap] Briscoe, B., Kaippallimalil, J., and P. Thaler, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP", Internet-draft, IETF work-in-progress draft-ietf-tsvwg-ecn-encap-guidelines.

[ID.Fallback] Kuehlewind, Mirja. and Brian. Trammell, "A Mechanism for ECN Path Probing and Fallback, draft-kuehlewind-tcpm-ecn-fallback, Work-in-Progress".

[RFC0768] Postel, J., "User Datagram Protocol", 1980.

[RFC1349] "Type of Service in the Internet Protocol Suite".

[RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, DOI 10.17487/RFC3649, December 2003, <http://www.rfc-editor.org/info/rfc3649>.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <http://www.rfc-editor.org/info/rfc3758>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <http://www.rfc-editor.org/info/rfc4340>.

[RFC4774] Floyd, S., "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field", BCP 124, RFC 4774, DOI 10.17487/RFC4774, November 2006, <http://www.rfc-editor.org/info/rfc4774>.

[RFC5562] Kuzmanovic, A., Mondal, A., Floyd, S., and K. Ramakrishnan, "Adding Explicit Congestion Notification (ECN) Capability to TCP’s SYN/ACK Packets", RFC 5562, DOI 10.17487/RFC5562, June 2009, <http://www.rfc-editor.org/info/rfc5562>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC6679] Westerlund, M., Johansson, I., Perkins, C., O’Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <http://www.rfc-editor.org/info/rfc6679>.

[RFC6789] Briscoe, B., Ed., Woundy, R., Ed., and A. Cooper, Ed., "Congestion Exposure (ConEx) Concepts and Use Cases", RFC 6789, DOI 10.17487/RFC6789, December 2012, <http://www.rfc-editor.org/info/rfc6789>.

[ST14] Stewart, R., Tuexen, M., and X. Dong, "ECN for Stream Control Transmission Protocol (SCTP)", Internet-draft draft-stewart-tsvwg-sctpecn-05.txt, January 2014.

[TR15] Tranmmel, Brian., Kuehlewind, Mirja., Boppart, Damiano, Learmonth, Iain., and Gorry. Fairhurst, "Enabling internet-wide deployment of Explicit Congestion Notification Tramwell, B., Kuehlewind, M., Boppart, D., Learmonth, I., Fairhurst, G. & Scheffnegger, Passive and Active Measurement Conference (PAM)", March 2015.

Authors’ Addresses

Godred Fairhurst
University of Aberdeen
School of Engineering, Fraser Noble Building
Aberdeen AB24 3UE
UK

Email: gorry@erg.abdn.ac.uk

Michael Welzl
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Phone: +47 22 85 24 20
Email: michawe@ifi.uio.no

DOTS
Internet-Draft
Intended status: Informational
Expires: September 24, 2019

A. Mortensen
Arbor Networks
T. Reddy
McAfee
R. Moskowitz
Huawei

March 23, 2019

Distributed Denial of Service (DDoS) Open Threat Signaling Requirements
draft-ietf-dots-requirements-22

Abstract

This document defines the requirements for the Distributed Denial of Service (DDoS) Open Threat Signaling (DOTS) protocols enabling coordinated response to DDoS attacks.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Context and Motivation

Distributed Denial of Service (DDoS) attacks afflict networks connected to the Internet, plaguing network operators at service providers and enterprises around the world. High-volume attacks saturating inbound links are now common, as attack scale and frequency continue to increase.

The prevalence and impact of these DDoS attacks has led to an increased focus on coordinated attack response. However, many enterprises lack the resources or expertise to operate on-premises attack mitigation solutions themselves, or are constrained by local bandwidth limitations. To address such gaps, service providers have begun to offer on-demand traffic scrubbing services, which are designed to separate the DDoS attack traffic from legitimate traffic and forward only the latter.

Today, these services offer proprietary interfaces for subscribers to request attack mitigation. Such proprietary interfaces tie a subscriber to a service and limit the abilities of network elements

that would otherwise be capable of participating in attack mitigation. As a result of signaling interface incompatibility, attack responses may be fragmented or otherwise incomplete, leaving operators in the attack path unable to assist in the defense.

A standardized method to coordinate a real-time response among involved operators will increase the speed and effectiveness of DDoS attack mitigation, and reduce the impact of these attacks. This document describes the required characteristics of protocols that enable attack response coordination and mitigation of DDoS attacks.

DDoS Open Threat Signaling (DOTS) communicates the need for defensive action in anticipation of or in response to an attack, but does not dictate the implementation of these actions. The DOTS use cases are discussed in [I-D.ietf-dots-use-cases] and the DOTS architecture is discussed in [I-D.ietf-dots-architecture].

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 [RFC2119] [RFC8174], when, and only when, they appear in all capitals. These capitalized words are used to signify the requirements for DOTS protocols design.

This document adopts the following terms:

DDoS: A distributed denial-of-service attack, in which traffic originating from multiple sources is directed at a target on a network. DDoS attacks are intended to cause a negative impact on the availability and/or other functionality of an attack target. Denial-of-service considerations are discussed in detail in [RFC4732].

DDoS attack target: A network connected entity that is the target of a DDoS attack. Potential targets include (but are not limited to) network elements, network links, servers, and services.

DDoS attack telemetry: Collected measurements and behavioral characteristics defining the nature of a DDoS attack.

Countermeasure: An action or set of actions focused on recognizing and filtering out specific types of DDoS attack traffic while passing legitimate traffic to the attack target. Distinct countermeasures can be layered to defend against attacks combining multiple DDoS attack types.

Mitigation: A set of countermeasures enforced against traffic destined for the target or targets of a detected or reported DDoS attack, where countermeasure enforcement is managed by an entity in the network path between attack sources and the attack target. Mitigation methodology is out of scope for this document.

Mitigator: An entity, typically a network element, capable of performing mitigation of a detected or reported DDoS attack. The means by which this entity performs these mitigations and how they are requested of it are out of scope for this document. The mitigator and DOTS server receiving a mitigation request are assumed to belong to the same administrative entity.

DOTS client: A DOTS-aware software module responsible for requesting attack response coordination with other DOTS-aware elements.

DOTS server: A DOTS-aware software module handling and responding to messages from DOTS clients. The DOTS server enables mitigation on behalf of the DOTS client, if requested, by communicating the DOTS client's request to the mitigator and returning selected mitigator feedback to the requesting DOTS client.

DOTS agent: Any DOTS-aware software module capable of participating in a DOTS signal or data channel. It can be a DOTS client, DOTS server, or, as a logical agent, a DOTS gateway.

DOTS gateway: A DOTS-aware software module resulting from the logical concatenation of the functionality of a DOTS server and a DOTS client into a single DOTS agent. This functionality is analogous to a Session Initiation Protocol (SIP) [RFC3261] Back-to-Back User Agent (B2BUA) [RFC7092]. A DOTS gateway has a client-facing side, which behaves as a DOTS server for downstream clients, and a server-facing side, which performs the role of DOTS client for upstream DOTS servers. Client-domain DOTS gateways are DOTS gateways that are in the DOTS client's domain, while server-domain DOTS gateways denote DOTS gateways that are in the DOTS server's domain. A DOTS gateway may terminate multiple discrete DOTS client connections and may aggregate these into a single or multiple connections. DOTS gateways are described further in [I-D.ietf-dots-architecture].

Signal channel: A bidirectional, mutually authenticated communication channel between DOTS agents that is resilient even in conditions leading to severe packet loss, such as a volumetric DDoS attack causing network congestion.

DOTS signal: A status/control message transmitted over the authenticated signal channel between DOTS agents, used to indicate

the client's need for mitigation, or to convey the status of any requested mitigation.

Heartbeat: A message transmitted between DOTS agents over the signal channel, used as a keep-alive and to measure peer health.

Data channel: A bidirectional, mutually authenticated communication channel between two DOTS agents used for infrequent but reliable bulk exchange of data not easily or appropriately communicated through the signal channel. Reliable bulk data exchange may not function well or at all during attacks causing network congestion. The data channel is not expected to operate in such conditions.

Filter: A specification of a matching network traffic flow or set of flows. The filter will typically have a policy associated with it, e.g., rate-limiting or discarding matching traffic [RFC4949].

Drop-list: A list of filters indicating sources from which traffic should be blocked, regardless of traffic content.

Accept-list: A list of filters indicating sources from which traffic should always be allowed, regardless of contradictory data gleaned in a detected attack.

Multi-homed DOTS client: A DOTS client exchanging messages with multiple DOTS servers, each in a separate administrative domain.

2. Requirements

The expected layout and interactions amongst DOTS entities is described in the DOTS Architecture [I-D.ietf-dots-architecture].

The goal of the DOTS requirements specification is to specify the requirements for DOTS signal channel and data channel protocols that have different application and transport layer requirements. This section describes the required features and characteristics of the DOTS protocols.

The goal of DOTS protocols is to enable and manage mitigation on behalf of a network domain or resource which is or may become the focus of a DDoS attack. An active DDoS attack against the entity controlling the DOTS client need not be present before establishing a communication channel between DOTS agents. Indeed, establishing a relationship with peer DOTS agents during normal network conditions provides the foundation for more rapid attack response against future attacks, as all interactions setting up DOTS, including any business or service level agreements, are already complete. Reachability information of peer DOTS agents is provisioned to a DOTS client using

a variety of manual or dynamic methods. Once a relationship between DOTS agents is established, regular communication between DOTS clients and servers enables a common understanding of the DOTS agents' health and activity.

The DOTS protocol must at a minimum make it possible for a DOTS client to request aid mounting a defense against a suspected attack. This defense could be coordinated by a DOTS server and include signaling within or between domains as requested by local operators. DOTS clients should similarly be able to withdraw aid requests. DOTS requires no justification from DOTS clients for requests for help, nor do DOTS clients need to justify withdrawing help requests: the decision is local to the DOTS clients' domain. Multi-homed DOTS clients must be able to select the appropriate DOTS server(s) to which a mitigation request is to be sent. The method for selecting the appropriate DOTS server in a multi-homed environment is out of scope for this document.

DOTS protocol implementations face competing operational goals when maintaining this bidirectional communication stream. On the one hand, DOTS must include measures to ensure message confidentiality, integrity, authenticity, and replay protection to keep the protocols from becoming additional vectors for the very attacks it is meant to help fight off. On the other hand, the protocol must be resilient under extremely hostile network conditions, providing continued contact between DOTS agents even as attack traffic saturates the link. Such resiliency may be developed several ways, but characteristics such as small message size, asynchronous, redundant message delivery and minimal connection overhead (when possible given local network policy) will tend to contribute to the robustness demanded by a viable DOTS protocol. Operators of peer DOTS-enabled domains may enable quality- or class-of-service traffic tagging to increase the probability of successful DOTS signal delivery, but DOTS does not require such policies be in place, and should be viable in their absence.

The DOTS server and client must also have some standardized method of defining the scope of any mitigation, as well as managing other mitigation-related configuration.

Finally, DOTS should be sufficiently extensible to meet future needs in coordinated attack defense, although this consideration is necessarily superseded by the other operational requirements.

2.1. General Requirements

GEN-001 Extensibility: Protocols and data models developed as part of DOTS MUST be extensible in order to keep DOTS adaptable to proprietary DDoS defenses. Future extensions MUST be backward compatible. Implementations of older protocol versions MUST ignore optional information added to DOTS messages as part of newer protocol versions. Implementations of older protocol versions MUST reject DOTS messages carrying mandatory information as part of newer protocol versions.

GEN-002 Resilience and Robustness: The signaling protocol MUST be designed to maximize the probability of signal delivery even under the severely constrained network conditions caused by attack traffic. Additional means to enhance the resilience of DOTS protocols, including when multiple DOTS servers are provisioned to the DOTS clients, SHOULD be considered. The protocol MUST be resilient, that is, continue operating despite message loss and out-of-order or redundant message delivery. In support of signaling protocol robustness, DOTS signals SHOULD be conveyed over transport and application protocols not susceptible to Head of Line Blocking. These requirements are at SHOULD strength to handle middle-boxes and firewall traversal.

GEN-003 Bulk Data Exchange: Infrequent bulk data exchange between DOTS agents can also significantly augment attack response coordination, permitting such tasks as population of drop- or accept-listed source addresses; address or prefix group aliasing; exchange of incident reports; and other hinting or configuration supplementing attack response.

As the resilience requirements for the DOTS signal channel mandate small signal message size, a separate, secure data channel utilizing a reliable transport protocol MUST be used for bulk data exchange. However, reliable bulk data exchange may not be possible during attacks causing network congestion.

GEN-004 Mitigation Hinting: DOTS clients may have access to attack details which can be used to inform mitigation techniques. Example attack details might include locally collected fingerprints for an on-going attack, or anticipated or active attack focal points based on other threat intelligence. DOTS clients MAY send mitigation hints derived from attack details to DOTS servers, in the full understanding that the DOTS server MAY ignore mitigation hints. Mitigation hints MUST be transmitted across the signal channel, as the data channel may not be functional during an attack. DOTS server handling of mitigation hints is implementation-specific.

GEN-005 Loop Handling: In certain scenarios, typically involving misconfiguration of DNS or routing policy, it may be possible for communication between DOTS agents to loop. Signal and data channel implementations should be prepared to detect and terminate such loops to prevent service disruption.

2.2. Signal Channel Requirements

SIG-001 Use of Common Transport Protocols: DOTS MUST operate over common widely deployed and standardized transport protocols. While connectionless transport such as the User Datagram Protocol (UDP) [RFC0768] SHOULD be used for the signal channel, the Transmission Control Protocol (TCP) [RFC0793] MAY be used if necessary due to network policy or middlebox capabilities or configurations.

SIG-002 Sub-MTU Message Size: To avoid message fragmentation and the consequently decreased probability of message delivery over a congested link, signaling protocol message size MUST be kept under signaling Path Maximum Transmission Unit (PMTU), including the byte overhead of any encapsulation, transport headers, and transport- or message-level security. If the total message size exceeds the path MTU, the DOTS agent MUST split the message into separate messages; for example, the list of mitigation scope types could be split into multiple lists and each list conveyed in a new message.

DOTS agents can attempt to learn PMTU using the procedures discussed in [I-D.ietf-intarea-frag-fragile]. If the PMTU cannot be discovered, DOTS agents MUST assume a PMTU of 1280 bytes, as IPv6 requires that every link in the Internet have an MTU of 1280 octets or greater as specified in [RFC8200]. If IPv4 support on legacy or otherwise unusual networks is a consideration and the PMTU is unknown, DOTS implementations MAY assume on a PMTU of 576 bytes for IPv4 datagrams, as every IPv4 host must be capable of receiving a packet whose length is equal to 576 bytes as discussed in [RFC0791] and [RFC1122].

SIG-003 Bidirectionality: To support peer health detection, to maintain an active signal channel, and to increase the probability of signal delivery during an attack, the signal channel MUST be bidirectional, with client and server transmitting signals to each other at regular intervals, regardless of any client request for mitigation. The bidirectional signal channel MUST support unidirectional messaging to enable notifications between DOTS agents.

SIG-004 Channel Health Monitoring: DOTS agents MUST support exchange of heartbeat messages over the signal channel to monitor channel health. These keepalives serve the purpose to maintain any on-path NAT or Firewall bindings to avoid cryptographic handshake for new mitigation requests. The heartbeat interval during active mitigation could be negotiable based on NAT/Firewall characteristics. Absent information about the NAT/Firewall characteristics, DOTS agent needs to ensure its on-path NAT or Firewall bindings do not expire, by using the keep-alive frequency discussed in Section 3.5 of [RFC8085].

To support scenarios in which loss of heartbeat is used to trigger mitigation, and to keep the channel active, DOTS servers MUST solicit heartbeat exchanges after successful mutual authentication. When DOTS agents are exchanging heartbeats and no mitigation request is active, either agent MAY request changes to the heartbeat rate. For example, a DOTS server might want to reduce heartbeat frequency or cease heartbeat exchanges when an active DOTS client has not requested mitigation, in order to control load.

Following mutual authentication, a signal channel MUST be considered active until a DOTS agent explicitly ends the session. When no attack traffic is present, the signal channel MUST be considered active until either DOTS agent fails to receive heartbeats from the other peer after a mutually agreed upon retransmission procedure has been exhausted. Peer DOTS agents MUST regularly send heartbeats to each other while a mitigation request is active. Because heartbeat loss is much more likely during volumetric attack, DOTS agents SHOULD avoid signal channel termination when mitigation is active and heartbeats are not received by either DOTS agent for an extended period. The exception circumstances to terminate the signal channel session during active mitigation are discussed below:

* To handle possible DOTS server restart or crash, the DOTS clients MAY attempt to establish a new signal channel session, but MUST continue to send heartbeats on the current session so that the DOTS server knows the session is still alive. If the new session is successfully established, the DOTS client can terminate the current session.

* DOTS servers are assumed to have the ability to monitor the attack, using feedback from the mitigator and other available sources, and MAY use the absence of attack traffic and lack of

client heartbeats as an indication the signal channel is defunct.

SIG-005 Channel Redirection: In order to increase DOTS operational flexibility and scalability, DOTS servers SHOULD be able to redirect DOTS clients to another DOTS server at any time. DOTS clients MUST NOT assume the redirection target DOTS server shares security state with the redirecting DOTS server. DOTS clients are free to attempt abbreviated security negotiation methods supported by the protocol, such as DTLS session resumption, but MUST be prepared to negotiate new security state with the redirection target DOTS server. The redirection DOTS server and redirecting DOTS server MUST belong to the same administrative domain.

Due to the increased likelihood of packet loss caused by link congestion during an attack, DOTS servers SHOULD NOT redirect while mitigation is enabled during an active attack against a target in the DOTS client's domain.

SIG-006 Mitigation Requests and Status: Authorized DOTS clients MUST be able to request scoped mitigation from DOTS servers. DOTS servers MUST send status to the DOTS clients about mitigation requests. If a DOTS server rejects an authorized request for mitigation, the DOTS server MUST include a reason for the rejection in the status message sent to the client.

DOTS servers MUST regularly send mitigation status updates to authorized DOTS clients which have requested and been granted mitigation. If unreliable transport is used for the signal channel protocol, due to the higher likelihood of packet loss during a DDoS attack, DOTS servers needs to send mitigation status multiple times at regular intervals following the the data transmission guidelines discussed in Section 3.1.3 of [RFC8085].

When DOTS client-requested mitigation is active, DOTS server status messages MUST include the following mitigation metrics:

* Total number of packets blocked by the mitigation

* Current number of packets per second blocked

* Total number of bytes blocked

* Current number of bytes per second blocked

DOTS clients MAY take these metrics into account when determining whether to ask the DOTS server to cease mitigation.

A DOTS client MAY withdraw a mitigation request at any time, regardless of whether mitigation is currently active. The DOTS server MUST immediately acknowledge a DOTS client's request to stop mitigation.

To protect against route or DNS flapping caused by a client rapidly toggling mitigation, and to dampen the effect of oscillating attacks, DOTS servers MAY allow mitigation to continue for a limited period after acknowledging a DOTS client's withdrawal of a mitigation request. During this period, DOTS server status messages SHOULD indicate that mitigation is active but terminating. DOTS clients MAY reverse the mitigation termination during this active-but-terminating period with a new mitigation request for the same scope. The DOTS server MUST treat this request as a mitigation lifetime extension (see SIG-007 below).

The initial active-but-terminating period is implementation- and deployment- specific, but SHOULD be sufficiently long to absorb latency incurred by route propagation. If a DOTS client refreshes the mitigation before the active-but-terminating period elapses, the DOTS server MAY increase the active-but-terminating period up to a maximum of 300 seconds (5 minutes). After the active-but-terminating period elapses, the DOTS server MUST treat the mitigation as terminated, as the DOTS client is no longer responsible for the mitigation.

SIG-007 Mitigation Lifetime: DOTS servers MUST support mitigations for a negotiated time interval, and MUST terminate a mitigation when the lifetime elapses. DOTS servers also MUST support renewal of mitigation lifetimes in mitigation requests from DOTS clients, allowing clients to extend mitigation as necessary for the duration of an attack.

DOTS servers MUST treat a mitigation terminated due to lifetime expiration exactly as if the DOTS client originating the mitigation had asked to end the mitigation, including the active-but-terminating period, as described above in SIG-005.

DOTS clients MUST include a mitigation lifetime in all mitigation requests.

DOTS servers SHOULD support indefinite mitigation lifetimes, enabling architectures in which the mitigator is always in the traffic path to the resources for which the DOTS client is requesting protection. DOTS clients MUST be prepared to not be granted mitigations with indefinite lifetimes. DOTS servers MAY refuse mitigations with indefinite lifetimes, for policy reasons.

The reasons themselves are out of scope. If the DOTS server does not grant a mitigation request with an indefinite mitigation lifetime, it MUST set the lifetime to a value that is configured locally. That value MUST be returned in a reply to the requesting DOTS client.

SIG-008 Mitigation Scope: DOTS clients MUST indicate desired mitigation scope. The scope type will vary depending on the resources requiring mitigation. All DOTS agent implementations MUST support the following required scope types:

* IPv4 prefixes [RFC4632]

* IPv6 prefixes [RFC4291][RFC5952]

* Domain names [RFC1035]

The following mitigation scope types are OPTIONAL:

* Uniform Resource Identifiers [RFC3986]

DOTS servers MUST be able to resolve domain names and (when supported) URIs. How name resolution is managed on the DOTS server is implementation-specific.

DOTS agents MUST support mitigation scope aliases, allowing DOTS clients and servers to refer to collections of protected resources by an opaque identifier created through the data channel, direct configuration, or other means. Domain name and URI mitigation scopes may be thought of as a form of scope alias, in which the addresses to which the domain name or URI resolve represent the full scope of the mitigation.

If there is additional information available narrowing the scope of any requested attack response, such as targeted port range, protocol, or service, DOTS clients SHOULD include that information in client mitigation requests. DOTS clients MAY also include additional attack details. DOTS servers MAY ignore such supplemental information when enabling countermeasures on the mitigator.

As an active attack evolves, DOTS clients MUST be able to adjust as necessary the scope of requested mitigation by refining the scope of resources requiring mitigation.

A DOTS client may obtain the mitigation scope through direct provisioning or through implementation-specific methods of

discovery. DOTS clients MUST support at least one mechanism to obtain mitigation scope.

SIG-009 Mitigation Efficacy: When a mitigation request is active, DOTS clients MUST be able to transmit a metric of perceived mitigation efficacy to the DOTS server. DOTS servers MAY use the efficacy metric to adjust countermeasures activated on a mitigator on behalf of a DOTS client.

SIG-010 Conflict Detection and Notification: Multiple DOTS clients controlled by a single administrative entity may send conflicting mitigation requests as a result of misconfiguration, operator error, or compromised DOTS clients. DOTS servers in the same administrative domain attempting to honor conflicting requests may flap network route or DNS information, degrading the networks attempting to participate in attack response with the DOTS clients. DOTS servers in a single administrative domain SHALL detect such conflicting requests, and SHALL notify the DOTS clients in conflict. The notification MUST indicate the nature and scope of the conflict, for example, the overlapping prefix range in a conflicting mitigation request.

SIG-011 Network Address Translator Traversal: DOTS clients may be deployed behind a Network Address Translator (NAT), and need to communicate with DOTS servers through the NAT. DOTS protocols MUST therefore be capable of traversing NATs.

If UDP is used as the transport for the DOTS signal channel, all considerations in "Middlebox Traversal Guidelines" in [RFC8085] apply to DOTS. Regardless of transport, DOTS protocols MUST follow established best common practices established in BCP 127 for NAT traversal [RFC4787][RFC6888][RFC7857].

2.3. Data Channel Requirements

The data channel is intended to be used for bulk data exchanges between DOTS agents. Unlike the signal channel, the data channel is not expected to be constructed to deal with attack conditions. As the primary function of the data channel is data exchange, a reliable transport is required in order for DOTS agents to detect data delivery success or failure.

The data channel provides a protocol for DOTS configuration, management. For example, a DOTS client may submit to a DOTS server a collection of prefixes it wants to refer to by alias when requesting mitigation, to which the server would respond with a success status and the new prefix group alias, or an error status and message in the event the DOTS client’s data channel request failed.

DATA-001 Reliable transport: Messages sent over the data channel MUST be delivered reliably, in order sent.

DATA-003 Resource Configuration: To help meet the general and signal channel requirements in Section 2.1 and Section 2.2, DOTS server implementations MUST provide an interface to configure resource identifiers, as described in SIG-008. DOTS server implementations MAY expose additional configurability. Additional configurability is implementation-specific.

DATA-004 Policy management: DOTS servers MUST provide methods for DOTS clients to manage drop- and accept-lists of traffic destined for resources belonging to a client.

For example, a DOTS client should be able to create a drop- or accept-list entry, retrieve a list of current entries from either list, update the content of either list, and delete entries as necessary.

How a DOTS server authorizes DOTS client management of drop- and accept-list entries is implementation-specific.

2.4. Security Requirements

DOTS must operate within a particularly strict security context, as an insufficiently protected signal or data channel may be subject to abuse, enabling or supplementing the very attacks DOTS purports to mitigate.

SEC-001 Peer Mutual Authentication: DOTS agents MUST authenticate each other before a DOTS signal or data channel is considered valid. The method of authentication is not specified in this document, but should follow current IETF best practices [RFC7525] with respect to any cryptographic mechanisms to authenticate the remote peer.

SEC-002 Message Confidentiality, Integrity and Authenticity: DOTS protocols MUST take steps to protect the confidentiality, integrity and authenticity of messages sent between client and server. While specific transport- and message-level security options are not specified, the protocols MUST follow current IETF best practices [RFC7525] for encryption and message authentication. Client-domain DOTS gateways are more trusted than DOTS clients, while server-domain DOTS gateways and DOTS servers share the same level of trust. A security mechanism at the transport layer TLS/DTLS is thus adequate to provide security between peer DOTS agents.

In order for DOTS protocols to remain secure despite advancements in cryptanalysis and traffic analysis, DOTS agents MUST support secure negotiation of the terms and mechanisms of protocol security, subject to the interoperability and signal message size requirements in Section 2.2.

While the interfaces between downstream DOTS server and upstream DOTS client within a DOTS gateway are implementation-specific, those interfaces nevertheless MUST provide security equivalent to that of the signal channels bridged by gateways in the signaling path. For example, when a DOTS gateway consisting of a DOTS server and DOTS client is running on the same logical device, the two DOTS agents could be implemented within the same process security boundary.

SEC-003 Data privacy and integrity: Transmissions over the DOTS protocols are likely to contain operationally or privacy-sensitive information or instructions from the remote DOTS agent. Theft, modification, or replay of message transmissions could lead to information leaks or malicious transactions on behalf of the sending agent (see Section 4 below). Consequently data sent over the DOTS protocols MUST be encrypted using secure transports TLS/DTLS. DOTS servers MUST enable means to prevent leaking operationally or privacy-sensitive data. Although administrative entities participating in DOTS may detail what data may be revealed to third-party DOTS agents, such considerations are not in scope for this document.

SEC-004 Message Replay Protection: To prevent a passive attacker from capturing and replaying old messages, and thereby potentially disrupting or influencing the network policy of the receiving DOTS agent's domain, DOTS protocols MUST provide a method for replay detection and prevention.

Within the signal channel, messages MUST be uniquely identified such that replayed or duplicated messages can be detected and discarded. Unique mitigation requests MUST be processed at most once.

SEC-005 Authorization: DOTS servers MUST authorize all messages from DOTS clients which pertain to mitigation, configuration, filtering, or status.

DOTS servers MUST reject mitigation requests with scopes which the DOTS client is not authorized to manage.

Likewise, DOTS servers MUST refuse to allow creation, modification or deletion of scope aliases and drop-/accept-lists when the DOTS client is unauthorized.

The modes of authorization are implementation-specific.

2.5. Data Model Requirements

A well-structured DOTS data model is critical to the development of successful DOTS protocols.

DM-001 Structure: The data model structure for the DOTS protocol MAY be described by a single module, or be divided into related collections of hierarchical modules and sub-modules. If the data model structure is split across modules, those distinct modules MUST allow references to describe the overall data model's structural dependencies.

DM-002 Versioning: To ensure interoperability between DOTS protocol implementations, data models MUST be versioned. How the protocols represent data model versions is not defined in this document.

DM-003 Mitigation Status Representation: The data model MUST provide the ability to represent a request for mitigation and the withdrawal of such a request. The data model MUST also support a representation of currently requested mitigation status, including failures and their causes.

DM-004 Mitigation Scope Representation: The data model MUST support representation of a requested mitigation's scope. As mitigation scope may be represented in several different ways, per SIG-008 above, the data model MUST include extensible representation of mitigation scope.

DM-005 Mitigation Lifetime Representation: The data model MUST support representation of a mitigation request's lifetime, including mitigations with no specified end time.

DM-006 Mitigation Efficacy Representation: The data model MUST support representation of a DOTS client's understanding of the efficacy of a mitigation enabled through a mitigation request.

DM-007 Acceptable Signal Loss Representation: The data model MUST be able to represent the DOTS agent's preference for acceptable signal loss when establishing a signal channel. Measurements of loss might include, but are not restricted to, number of consecutive missed heartbeat messages, retransmission count, or request timeouts.

DM-008 Heartbeat Interval Representation: The data model MUST be able to represent the DOTS agent's preferred heartbeat interval, which the client may include when establishing the signal channel, as described in SIG-003.

DM-009 Relationship to Transport: The DOTS data model MUST NOT make any assumptions about specific characteristics of any given transport into the data model, but instead, represent the fields in the model explicitly.

3. Congestion Control Considerations

3.1. Signal Channel

As part of a protocol expected to operate over links affected by DDoS attack traffic, the DOTS signal channel MUST NOT contribute significantly to link congestion. To meet the signal channel requirements above, DOTS signal channel implementations SHOULD support connectionless transports. However, some connectionless transports when deployed naively can be a source of network congestion, as discussed in [RFC8085]. Signal channel implementations using such connectionless transports, such as UDP, therefore MUST include a congestion control mechanism.

Signal channel implementations using a IETF standard congestion-controlled transport protocol (like TCP) may rely on built-in transport congestion control support.

3.2. Data Channel

As specified in DATA-001, the data channel requires reliable, in-order message delivery. Data channel implementations using a IETF standard congestion-controlled transport protocol may rely on the transport implementation's built-in congestion control mechanisms.

4. Security Considerations

This document informs future protocols under development, and so does not have security considerations of its own. However, operators should be aware of potential risks involved in deploying DOTS. DOTS agent impersonation and signal blocking are discussed here. Additional DOTS security considerations may be found in [I-D.ietf-dots-architecture] and DOTS protocol documents.

Impersonation of either a DOTS server or a DOTS client could have catastrophic impact on operations in either domain. If an attacker has the ability to impersonate a DOTS client, that attacker can affect policy on the network path to the DOTS client's domain, up to

and including instantiation of drop-lists blocking all inbound traffic to networks for which the DOTS client is authorized to request mitigation.

Similarly, an impersonated DOTS server may be able to act as a sort of malicious DOTS gateway, intercepting requests from the downstream DOTS client, and modifying them before transmission to the DOTS server to inflict the desired impact on traffic to or from the DOTS client's domain. Among other things, this malicious DOTS gateway might receive and discard mitigation requests from the DOTS client, ensuring no requested mitigation is ever applied.

To detect misuse, as detailed in Section 2.4, DOTS implementations require mutual authentication of DOTS agents in order to make agent impersonation more difficult. However, impersonation may still be possible as a result of credential theft, implementation flaws, or compromise of DOTS agents.

To detect compromised DOTS agents, DOTS operators should carefully monitor and audit DOTS agents to detect misbehavior and to deter misuse, while employing current secure network communications best practices to reduce attack surface.

Blocking communication between DOTS agents has the potential to disrupt the core function of DOTS, which is to request mitigation of active or expected DDoS attacks. The DOTS signal channel is expected to operate over congested inbound links, and, as described in Section 2.2, the signal channel protocol must be designed for minimal data transfer to reduce the incidence of signal loss.

5. IANA Considerations

This document does not require any IANA action.

6. Contributors

Mohamed Boucadair
Orange

mohamed.boucadair@orange.com

Flemming Andreasen
Cisco Systems, Inc.

fandreas@cisco.com

Dave Dolson
Sandvine

ddolson@sandvine.com

7. Acknowledgments

Thanks to Roman Danyliw, Matt Richardson, Joe Touch, Scott Bradner, Robert Sparks, Brian Weis, Benjamin Kaduk, Eric Rescorla, Alvaro Retana, Suresh Krishnan, Ben Campbell, Mirja Kuehlewind and Jon Shallow for careful reading and feedback.

8. References

8.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <https://www.rfc-editor.org/info/rfc4632>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <https://www.rfc-editor.org/info/rfc4787>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888, April 2013, <https://www.rfc-editor.org/info/rfc6888>.

[RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar, S., and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements", BCP 127, RFC 7857, DOI 10.17487/RFC7857, April 2016, <https://www.rfc-editor.org/info/rfc7857>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

8.2. Informative References

[I-D.ietf-dots-architecture]
Mortensen, A., Andreasen, F., K, R., Teague, N., Compton, R., and c. christopher_gray3@cable.comcast.com, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-12 (work in progress), February 2019.

[I-D.ietf-dots-use-cases]
Dobbins, R., Migault, D., Fouant, S., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-17 (work in progress), January 2019.

[I-D.ietf-intarea-frag-fragile]
Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", draft-ietf-intarea-frag-fragile-09 (work in progress), February 2019.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session Initiation Protocol (SIP) Back-to-Back User Agents", RFC 7092, DOI 10.17487/RFC7092, December 2013, <https://www.rfc-editor.org/info/rfc7092>.

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, DOI 10.17487/RFC4732, December 2006, <https://www.rfc-editor.org/info/rfc4732>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

Authors' Addresses

Andrew Mortensen
Arbor Networks
2727 S. State St
Ann Arbor, MI 48104
United States

Email: andrewmortensen@gmail.com

Tirumaleswar Reddy
McAfee
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: TirumaleswarReddy_Konda@McAfee.com

Robert Moskowitz
Huawei
Oak Park, MI 42837
United States

Email: rgm@htt-consult.com

DOTS
Internet-Draft
Intended status: Informational
Expires: January 6, 2021

R. Dobbins
Arbor Networks
D. Migault
Ericsson
R. Moskowitz
HTT Consulting
N. Teague
Iron Mountain Data Centers
L. Xia
Huawei
K. Nishizuka
NTT Communications
July 05, 2020

Use cases for DDoS Open Threat Signaling
draft-ietf-dots-use-cases-25

Abstract

The DDoS Open Threat Signaling (DOTS) effort is intended to provide protocols to facilitate interoperability across disparate DDoS mitigation solutions. This document presents sample use cases which describe the interactions expected between the DOTS components as well as DOTS messaging exchanges. These use cases are meant to identify the interacting DOTS components, how they collaborate, and what are the typical information to be exchanged.

Status of This Memo

Table of Contents

1. Introduction

At the time of writing, distributed denial-of-service (DDoS) attack mitigation solutions are largely based upon siloed, proprietary communications schemes with vendor lock-in as a side-effect. This can result in the configuration, provisioning, operation, and activation of these solutions being a highly manual and often time-consuming process. Additionally, coordinating multiple DDoS mitigation solutions simultaneously is fraught with both technical and process-related hurdles. This greatly increases operational complexity which, in turn, can degrade the efficacy of mitigations that are generally highly dependent on a timely reaction by the system.

The DDoS Open Threat Signaling (DOTS) effort is intended to specify protocols that facilitate interoperability between diverse DDoS

mitigation solutions and ensure greater integration in term of attack detection, mitigation requests, and attack characterization patterns.

As DDoS solutions are broadly heterogeneous among vendors, the primary goal of DOTS is to provide high-level interaction amongst differing DDoS solutions, such as detecting DDoS attacks, initiating/terminating DDoS mitigation assistance, or requesting the status of a DDoS mitigation.

This document provides sample use cases that provided input for the requirements [RFC8612] and design of the DOTS protocols [RFC8782][RFC8783]. The use cases are not exhaustive and future use cases are expected to emerge as DOTS is adopted and evolves.

2. Terminology and Acronyms

This document makes use of the same terminology and definitions as [RFC8612]. In addition it uses the terms defined below:

- o DDoS Mitigation System (DMS): A system that performs DDoS mitigation. The DDoS Mitigation System may be composed of a cluster of hardware and/or software resources, but could also involve an orchestrator that may take decisions such as outsourcing some or all of the mitigation to another DDoS Mitigation System.

- o DDoS Mitigation: The action performed by the DDoS Mitigation System.

- o DDoS Mitigation Service: designates a service provided to a customer to mitigate DDoS attacks. Each service subscription usually involve Service Level Agreement (SLA) that has to be met. It is the responsibility of the DDoS Service provider to instantiate the DDoS Mitigation System to meet these SLAs.

- o DDoS Mitigation Service Provider: designates the administrative entity providing the DDoS Mitigation Service.

- o Internet Transit Provider (ITP): designates the entity that delivers the traffic to a customer network. It can be an Internet Service Provider (ISP), or an upstream entity delivering the traffic to the ISP.

3. Use Cases

3.1. Upstream DDoS Mitigation by an Upstream Internet Transit Provider

This use case describes how an enterprise or a residential customer network may take advantage of a pre-existing relation with its ITP in order to mitigate a DDoS attack targeting its network.

For clarity of discussion, the targeted network is indicated as an enterprise network, but the same scenario applies to any downstream network, including residential and cloud hosting networks.

As the ITP provides connectivity to the enterprise network, it is already on the path of the inbound and outbound traffic of the enterprise network and well aware of the networking parameters associated to the enterprise network WAN connectivity. This eases both the configuration and the instantiation of a DDoS Mitigation Service.

This section considers two kinds of DDoS Mitigation Service between an enterprise network and an ITP:

- The upstream ITP may instantiate a DDoS Mitigation System (DMS) upon receiving a request from the enterprise network. This typically corresponds to the case when the enterprise network is under attack.

- On the other hand, the ITP may identify an enterprise network as the source of an attack and send a mitigation request to the enterprise DMS to mitigate this at the source.

The two scenarios, though different, have similar interactions between the DOTS client and server. For the sake of simplicity, only the first scenario will be detailed in this section. Nevertheless, the second scenario is also in scope for DOTS.

In the first scenario, as depicted in Figure 1, an enterprise network with self-hosted Internet-facing properties such as Web servers, authoritative DNS servers, and VoIP servers has a DMS deployed to protect those servers and applications from DDoS attacks. In addition to on-premise DDoS defense capability, the enterprise has contracted with its ITP for DDoS Mitigation Services when attacks threaten to overwhelm the bandwidth of their WAN link(s).

```
      +------------------+        +------------------+
      | Enterprise       |        | Upstream         |
      | Network          |        | Internet Transit |
      |                  |        | Provider         |
      |      +--------+  |        |             DDoS Attack
      |      | DDoS   |  | <================================
      |      | Target |  | <================================
      |      +--------+  |        |  +------------+  |
      |                  | +-------->| DDoS       |  |
      |                  | |      |S | Mitigation |  |
      |                  | |      |  | System     |  |
      |                  | |      |  +------------+  |
      |                  | |      |                  |
      |                  | |      |                  |
      |                  | |      |                  |
      |  +------------+  | |      |                  |
      |  | DDoS       |<---+      |                  |
      |  | Mitigation |C |        |                  |
      |  | System     |  |        |                  |
      |  +------------+  |        |                  |
      +------------------+        +------------------+

         * C is for DOTS client functionality
         * S is for DOTS server functionality
```

Figure 1: Upstream Internet Transit Provider DDoS Mitigation

The enterprise DMS is configured such that if the incoming Internet traffic volume exceeds 50% of the provisioned upstream Internet WAN link capacity, the DMS will request DDoS mitigation assistance from the upstream transit provider. More sophisticated detection means may be considered as well.

The requests to trigger, manage, and finalize a DDoS Mitigation between the enterprise DMS and the ITP is performed using DOTS. The enterprise DMS implements a DOTS client while the ITP implements a DOTS server which is integrated with their DMS in this example.

When the enterprise DMS locally detects an inbound DDoS attack targeting its resources (e.g., servers, hosts, or applications), it immediately begins a DDoS Mitigation.

During the course of the attack, the inbound traffic volume to the enterprise network exceeds the 50% threshold and the enterprise DMS escalates the DDoS mitigation. The enterprise DMS DOTS client signals to the DOTS server on the upstream ITP to initiate DDoS Mitigation. The DOTS server replies to the DOTS client that it can

serve this request, and mitigation is initiated on the ITP network by the ITP DMS.

Over the course of the attack, the DOTS server of the ITP periodically informs the DOTS client on the mitigation status, statistics related to DDoS attack traffic mitigation, and related information. Once the DDoS attack has ended, or decreased to a certain level that the enterprise DMS might handle by itself, the DOTS server signals the enterprise DMS DOTS client that the attack has subsided.

The DOTS client on the enterprise DMS then requests the ITP to terminate the DDoS Mitigation. The DOTS server on the ITP receives this request and once the mitigation has ended, confirms the end of upstream DDoS Mitigation to the enterprise DMS DOTS client.

The following is an overview of the DOTS communication model for this use-case:

1. A DDoS attack is initiated against resources of a network organization (here, the enterprise) which has deployed a DOTS-capable DMS - typically a DOTS client.

2. The enterprise DMS detects, classifies, and begins the DDoS Mitigation.

3. The enterprise DMS determines that its capacity and/or capability to mitigate the DDoS attack is insufficient, and sends via its DOTS client a DOTS DDoS Mitigation request to one or more DOTS servers residing on the upstream ITP.

4. The DOTS server which receives the DOTS Mitigation request determines that it has been configured to honor requests from the requesting DOTS client, and honors the request by orchestrating its own DMS.

5. While the DDoS Mitigation is active, the DOTS server regularly transmits DOTS DDoS Mitigation status updates to the DOTS client.

6. Informed by the DOTS server status update that the attack has ended or subsided, the DOTS client transmits a DOTS DDoS Mitigation termination request to the DOTS server.

7. The DOTS server terminates DDoS Mitigation, and sends the notification to the DOTS client.

Note that communications between the enterprise DOTS client and the upstream ITP DOTS server may take place in-band within the main

Internet WAN link between the enterprise and the ITP; out-of-band via a separate, dedicated wireline network link utilized solely for DOTS signaling; or out-of-band via some other form of network connectivity such as a third-party wireless 4G network connectivity.

Note also that a DOTS client that sends a DOTS Mitigation request may be also triggered by a network admin that manually confirms the request to the upstream ITP, in which case the request may be sent from an application such as a web browser or a dedicated mobile application.

Note also that when the enterprise is multihomed and connected to multiple upstream ITPs, each ITP is only able to provide a DDoS Mitigation Service for the traffic it transits. As a result, the enterprise network may be required to coordinate the various DDoS Mitigation Services associated to each link. More multi-homing considerations are discussed in [I-D.ietf-dots-multihoming].

3.2. DDoS Mitigation by a Third Party DDoS Mitigation Service Provider

This use case differs from the previous use case described in Section 3.1 in that the DDoS Mitigation Service is not provided by an upstream ITP. In other words, as represented in Figure 2, the traffic is not forwarded through the DDoS Mitigation Service Provider by default. In order to steer the traffic to the DDoS Mitigation Service Provider, some network configuration changes are required. As such, this use case is likely to apply to large enterprises or large data centers, but as for the other use cases is not exclusively limited to them.

Another typical scenario for this use case is for there to be a relationship between DDoS Mitigation Service Providers, forming an overlay of DMS. When a DDoS Mitigation Service Provider mitigating a DDoS attack reaches its resources capacity, it may chose to delegate the DDoS Mitigation to another DDoS Mitigation Service Provider.

```
      +------------------+        +------------------+
      | Enterprise       |        | Upstream         |
      | Network          |        | Internet Transit |
      |                  |        | Provider         |
      |      +--------+  |        |             DDoS Attack
      |      | DDoS   |  | <==================================
      |      | Target |  | <==================================
      |      +--------+  |        |                  |
      |                  |        |                  |
      |                  |        +------------------+
      |                  |
      |                  |        +------------------+
      |                  |        | DDoS Mitigation  |
      |                  |        | Service Provider |
      |                  |        |                  |
      |  +------------+  |        |  +------------+  |
      |  | DDoS       |<------------>| DDoS       |  |
      |  | Mitigation |C |        | S| Mitigation |  |
      |  | System     |  |        |  | System     |  |
      |  +------------+  |        |  +------------+  |
      +------------------+        +------------------+

          * C is for DOTS client functionality
          * S is for DOTS server functionality
```

Figure 2: DDoS Mitigation between an Enterprise Network and Third Party DDoS Mitigation Service Provider

In this scenario, an enterprise network has entered into a pre-arranged DDoS mitigation assistance agreement with one or more third-party DDoS Mitigation Service Providers in order to ensure that sufficient DDoS mitigation capacity and/or capabilities may be activated in the event that a given DDoS attack threatens to overwhelm the ability of the enterprise's or any other given DMS to mitigate the attack on its own.

The pre-arrangement typically includes agreement on the mechanisms used to redirect the traffic to the DDoS Mitigation Service Provider, as well as the mechanism to re-inject the traffic back to the Enterprise Network. Redirection to the DDoS Mitigation Service Provider typically involves BGP prefix announcement or DNS redirection, while re-injection of the scrubbed traffic to the enterprise network may be performed via tunneling mechanisms (e.g., GRE). The exact mechanisms used for traffic steering are out of scope of DOTS, but will need to be pre-arranged, while in some contexts such changes could be detected and considered as an attack.

In some cases the communication between the enterprise DOTS client and the DOTS server of the DDoS Mitigation Service Provider may go through the ITP carrying the DDoS attack, which would affect the communication. On the other hand, the communication between the DOTS client and DOTS server may take a path that is not undergoing a DDoS attack.

```
  +------------------+        +------------------+
  | Enterprise       |        | Upstream         |
  | Network          |        | Internet Transit |
  |                  |        | Provider         |
  |      +--------+  |        |              DDoS Attack
  |      | DDoS   |  |<----------------+         | ++====
  |      | Target |  |    Mitigated    |         | || ++=
  |      +--------+  |        |        |         | || ||
  |                  |        |        |         | || ||
  |                  |        +--------|---------+ || ||
  |                  |                 |           || ||
  |                  |        +--------|---------+ || ||
  |                  |        | DDoS Mitigation  | || ||
  |                  |        | Service Provider | || ||
  |                  |        |        |         | || ||
  |  +------------+  |        |  +------------+  | || ||
  |  | DDoS       |<------------>| DDoS       |  | || ||
  |  | mitigation |C |        |S | mitigation |<===++ ||
  |  | system     |  |        |  | system     |<======++
  |  +------------+  |        |  +------------+  |
  +------------------+        +------------------+

       * C is for DOTS client functionality
       * S is for DOTS server functionality
```

Figure 3: Redirection to a DDoS Mitigation Service Provider

When the enterprise network is under attack or at least is reaching its capacity or ability to mitigate a given DDoS attack, the DOTS client sends a DOTS request to the DDoS Mitigation Service Provider to initiate network traffic diversion - as represented in Figure 3 - and DDoS mitigation activities. Ongoing attack and mitigation status messages may be passed between the enterprise network and the DDoS Mitigation Service Provider using DOTS. If the DDoS attack has stopped or the severity of the attack has subsided, the DOTS client can request the DDoS Mitigation Service Provider to terminate the DDoS Mitigation.

3.3. DDoS Orchestration

In this use case, one or more DDoS telemetry systems or monitoring devices monitor a network - typically an ISP network, an enterprise network, or a data center. Upon detection of a DDoS attack, these DDoS telemetry systems alert an orchestrator in charge of coordinating the various DMS's within the domain. The DDoS telemetry systems may be configured to provide required information, such as a preliminary analysis of the observation, to the orchestrator.

The orchestrator analyses the various sets of information it receives from DDoS telemetry systems, and initiates one or more DDoS mitigation strategies. For example, the orchestrator could select the DDoS mitigation system in the enterprise network or one provided by the ITP.

DDoS Mitigation System selection and DDoS Mitigation techniques may depend on the type of the DDoS attack. In some case, a manual confirmation or selection may also be required to choose a proposed strategy to initiate a DDoS Mitigation. The DDoS Mitigation may consist of multiple steps such as configuring the network, or of updating already instantiated DDoS mitigation functions. Eventually, the coordination of the mitigation may involve external DDoS mitigation resources such as a transit provider or a Third Party DDoS Mitigation Service Provider.

The communication used to trigger a DDoS Mitigation between the DDoS telemetry and monitoring systems and the orchestrator is performed using DOTS. The DDoS telemetry system implements a DOTS client while the orchestrator implements a DOTS server.

The communication between a network administrator and the orchestrator is also performed using DOTS. The network administrator uses, for example, a web interface which interacts with a DOTS client, while the orchestrator implements a DOTS server.

The communication between the orchestrator and the DDoS Mitigation Systems is performed using DOTS. The orchestrator implements a DOTS client while the DDoS Mitigation Systems implement a DOTS server.

The configuration aspects of each DDoS Mitigation System, as well as the instantiations of DDoS mitigation functions or network configuration is not part of DOTS. Similarly, the discovery of available DDoS mitigation functions is not part of DOTS; and as such is out of scope.

```
          +----------+
          | network  |C            (Enterprise Network)
          | adminis  |<-+
          | trator   |  |
          +----------+  |
                        |
          +----------+  | S+--------------+     +-----------+
          |telemetry/|  +->|               |C   S| DDoS      |+
          |monitoring|<--->| Orchestrator  |<--->| mitigation||
          |systems   |C   S|               |<-+  | systems   ||
          +----------+     +--------------+C |  +-----------+|
                                              |    +----------+
          ------------------------------------|-----------------
                                              |
                                              |
             (Internet Transit Provider)      |
                                              |  +-----------+
                                              | S| DDoS      |+
                                             +->| mitigation||
                                                | systems   ||
                                                +-----------+|
          * C is for DOTS client functionality    +----------+
          * S is for DOTS server functionality
```

Figure 4: DDoS Orchestration

The DDoS telemetry systems monitor various aspects of the network traffic and perform some measurement tasks.

These systems are configured so that when an event or some measurement indicators reach a predefined level their associated DOTS client sends a DOTS mitigation request to the orchestrator DOTS server. The DOTS mitigation request may be associated with some optional mitigation hints to let the orchestrator know what has triggered the request. In particular, it is possible for something that locally to one telemetry system looks like an attack is not actually an attack when seen from the broader scope (e.g., of the orchestrator)

Upon receipt of the DOTS mitigation request from the DDoS telemetry system, the orchestrator DOTS server responds with an acknowledgment, to avoid retransmission of the request for mitigation. The orchestrator may begin collecting additional fine-grained and specific information from various DDoS telemetry systems in order to correlate the measurements and provide an analysis of the event. Eventually, the orchestrator may ask for additional information from the DDoS telemetry system; however, the collection of this information is out of scope of DOTS.

The orchestrator may be configured to start a DDoS Mitigation upon approval from a network administrator. The analysis from the orchestrator is reported to the network administrator via, for example, a web interface. If the network administrator decides to start the mitigation, the network administrator triggers the DDoS mitigation request using, for example, a web interface of a DOTS client communicating to the orchestrator DOTS server. This request is expected to be associated with a context that provides sufficient information to the orchestrator DOTS server to infer, elaborate and coordinate the appropriate DDoS Mitigation.

Upon receiving a request to mitigate a DDoS attack aimed at a target, the orchestrator may evaluate the volume of the attack as well as the value that the target represents. The orchestrator may select the DDoS Mitigation Service Provider based on the attack severity. It may also coordinate the DDoS Mitigation performed by the DDoS Mitigation Service Provider with some other tasks such as, for example, moving the target to another network so new sessions will not be impacted. The orchestrator requests a DDoS Mitigation by the selected DDoS mitigation systems via its DOTS client, as described in Section 3.1.

The orchestrator DOTS client is notified that the DDoS Mitigation is effective by the selected DDoS mitigation systems. The orchestrator DOTS server returns this information back to the network administrator.

Similarly, when the DDoS attack has stopped, the orchestrator DOTS client is notified and the orchestrator's DOTS server indicates to the DDoS telemetry systems as well as to the network administrator the end of the DDoS Mitigation.

In addition to the above DDoS Orchestration, the selected DDoS mitigation system can return back a mitigation request to the orchestrator as an offloading. For example, when the DDoS attack becomes severe and the DDoS mitigation system's utilization rate reaches its maximum capacity, the DDoS mitigation system can send mitigation requests with additional hints such as its blocked traffic information to the orchestrator. Then the orchestrator can take further actions such as requesting forwarding nodes such as routers to filter the traffic. In this case, the DDoS mitigation system implements a DOTS client while the orchestrator implements a DOTS server. Similar to other DOTS use cases, the offloading scenario assumes that some validation checks are followed by the DMS, the orchestrator, or both (e.g., avoid exhausting the resources of the forwarding nodes or inadvertent disruption of legitimate services). These validation checks are part of the mitigation, and are therefore out of the scope of the document.

4. Security Considerations

The document does not describe any protocol, though there are still a few high-level security considerations to discuss.

DOTS is at risk from three primary attacks: DOTS agent impersonation, traffic injection, and signaling blocking.

Impersonation and traffic injection mitigation can be mitigated through current secure communications best practices including mutual authentication. Preconfigured mitigation steps to take on the loss of keepalive traffic can partially mitigate signal blocking, but in general it is impossible to comprehensively defend against an attacker that can selectively block any or all traffic. Alternate communication paths that are (hopefully) not subject to blocking by the attacker in question is another potential mitigation.

Additional details of DOTS security requirements can be found in [RFC8612].

Service disruption may be experienced if inadequate mitigation actions are applied. These considerations are out of the scope of DOTS.

5. IANA Considerations

No IANA considerations exist for this document.

6. Acknowledgments

The authors would like to thank among others Tirumaleswar Reddy; Andrew Mortensen; Mohamed Boucadair; Artyom Gavrichenkov; Jon Shallow, Yuuhei Hayashi, Elwyn Davies, the DOTS WG chairs, Roman Danyliw and Tobias Gondrom as well as the Security AD Benjamin Kaduk for their valuable feedback.

We also would like to thank Stephan Fouant that was part of the initial co-authors of the documents.

7. Informative References

[I-D.ietf-dots-multihoming]
Boucadair, M., Reddy.K, T., and W. Pan, "Multi-homing Deployment Considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS)", draft-ietf-dots-multihoming-04 (work in progress), May 2020.

[RFC8612] Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open Threat Signaling (DOTS) Requirements", RFC 8612, DOI 10.17487/RFC8612, May 2019, <https://www.rfc-editor.org/info/rfc8612>.

[RFC8782] Reddy.K, T., Ed., Boucadair, M., Ed., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 8782, DOI 10.17487/RFC8782, May 2020, <https://www.rfc-editor.org/info/rfc8782>.

[RFC8783] Boucadair, M., Ed. and T. Reddy.K, Ed., "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", RFC 8783, DOI 10.17487/RFC8783, May 2020, <https://www.rfc-editor.org/info/rfc8783>.

Authors' Addresses

Roland Dobbins
Arbor Networks
Singapore

EMail: rdobbins@arbor.net

Daniel Migault
Ericsson
8275 Trans Canada Route
Saint Laurent, QC 4S 0B6
Canada

EMail: daniel.migault@ericsson.com

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
USA

EMail: rgm@labs.htt-consult.com

Nik Teague
Iron Mountain Data Centers
UK

EMail: nteague@ironmountain.co.uk

Liang Xia
Huawei
No. 101, Software Avenue, Yuhuatai District
Nanjing
China

EMail: Frank.xialiang@huawei.com

Kaname Nishizuka
NTT Communications
GranPark 16F 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

EMail: kaname@nttv6.jp

Internet Area WG
Internet Draft
Intended status: Informational
Expires: January 2016

J. Touch
USC/ISI
M. Townsley
Cisco

July 20, 2015

IP Tunnels in the Internet Architecture
draft-ietf-intarea-tunnels-01.txt

Status of this Memo

Copyright Notice

Abstract

This document discusses the role of IP tunnels in the Internet architecture. It explains their relationship to existing protocol layers and the challenges in supporting IP tunneling based on the equivalence of tunnels to links.

Table of Contents

1. Introduction

The Internet is loosely based on the ISO seven layer stack, in which data units traverse the stack by being wrapped inside data units one layer down. A tunnel is a mechanism for transmitting data units between endpoints by wrapping them as data units of the same or higher layers, e.g., IP in IP (Figure 1) or IP in UDP (Figure 2).

```
+----+----+--------------+
| IP'| IP |     Data     |
+----+----+--------------+
```

Figure 1 IP inside IP

```
+----+-----+----+--------------+
| IP'| UDP | IP |     Data     |
+----+-----+----+--------------+
```

Figure 2 IP in UDP in IP in Ethernet

This document focuses on tunnels that transit IP packets, i.e., in which an IP packet is the payload of another protocol. Tunnels provide a virtual link that can help decouple the network topology seen by transiting packets from the underlying physical network [To98][RFC2473]. For example, tunnels were critical in the development of multicast because not all routers were capable of processing multicast packets [Er94]. Tunnels allowed multicast packets to transit between multicast-capable routers over paths that did not support multicast. Similar techniques have been used to support other protocols, such as IPv6 [RFC2460].

Use of tunnels is common in the Internet. The word "tunnel" occurs in over 100 RFCs, and is supported within numerous protocols, including:

- Generic UDP Encapsulation (GUE) - IP in UDP (in IP)[He15a][He15b]

- Generic IPv6 tunneling [RFC2473]

- Generic Router Encapsulation (GRE) - an encapsulation framework allowing different messages to tunnel over a variety of tunnels, e.g., IP in GRE in IP [RFC2473][RFC2784][RFC7588][Pi15]

- IP in IP / mobile IP [RFC2003][RFC2473][RFC5944]

- IPsec - hides the original traffic destination [RFC4301]

- L2TP - Tunnels PPP over IP, used largely in DSL/FTTH access networks to extend a subscriber's connection from an access line provider to an ISP [RFC3931]

- L2VPNs - provides a link topology different from that provided by physical links [RFC4664]

- L3VPNs - provides a network topology different from that provided by ISPs [RFC4176]

- LISP - reduces routing table load within an enclave of routers [RFC6830]

- o MPLS - tunnels IP over a circuit-like path in which identifiers are rewritten on each hop, often used for traffic provisioning [RFC3031]

- o NVO3 - tunnels for data center network sharing (which includes use of GUE, above) [RFC7364]

- o PWE3 - tunnels to emulate wire-like services over packet-switched services [RFC3985]

- o SEAL/AERO - a generic mechanism for IP in IP tunneling designed to overcome the limitations of RFC2003 [RFC5320][Te15]

- o TRILL - enables L3 routing (typically IS-IS) in an enclave of Ethernet bridges [RFC5556][RFC6325]

The variety of tunnel mechanisms raises the question of the role of tunnels in the Internet architecture and the potential need for these mechanisms to have similar and predictable behavior. In particular, the ways in which packet sizes (i.e., Maximum Transmission Unit or MTU) mismatch and error signals (e.g., ICMP) are handled may benefit from a coordinated approach.

It is useful to note that, regardless of the layer in which encapsulation occurs, tunnels emulate a link. As links, they are subject to link issues, e.g., MTU discovery, signaling, and the potential utility of native support for broadcast and multicast [RFC2460][RFC3819]. They have advantages over native links, being potentially easier to reconfigure and control.

The remainder of this document describes the general principles of IP tunneling and discusses the key considerations in the design of a protocol that tunnels IP datagrams. It derives its conclusions from the equivalence of tunnels and links. Note that all considerations are in the context of existing standards and requirements.

2. Conventions used in this document

2.1. Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

2.2. Terminology

This document uses the following terminology. These definitions are given in the most general terms, but will be used primarily to discuss IP tunnels in this document. They are presented in order from most fundamental to those derived on earlier definitions:

- o Messages: variable length data labeled with globally-unique endpoint IDs [RFC791]

- o Endpoint: a network device that sources or sinks messages labeled from/to its IDs, also known as a host [RFC1122].

- o Forwarder: a network device that relays IP messages using longest-prefix match of destination IDs and local context, when possible, also known as a gateway or router [RFC1812].

- o Network node (node): an endpoint or forwarder. For Internet messages (IP datagrams), these are hosts or gateways/routers, respectively.

- o Source: the origin host of a message.

- o Destination: the receiving host of a message.

- o Link: a communication device that transfers messages between network devices, i.e., by which a message can traverse between devices without being processed by a forwarder. Note that the notion of forwarder is relative to the layer at which message processing is considered [RFC1122][RFC1812].

- o Path: a communications path by which a message can traverse between network nodes, which may or may not involve being processed by a forwarding node.

- o Tunnel: a protocol mechanism that transits messages using encapsulation to allow a path to appear as a link. Note that a protocol can be used to tunnel itself (IP over IP) and that this includes the conventional layering of the ISO stack (i.e., by this definition, Ethernet is a tunnel for IP).

- o Ingress: a network node that receives messages, encapsulates them according to the tunnel protocol, and transmits them into the tunnel. Note that the ingress and source can be co-located.

- o Egress: a network node that receives messages that have finished transiting a tunnel. The egress decapsulates datagrams for further transit to the destination. Note that the egress and destination can be co-located.

- o Tunnel transit packet: the packet arriving at a node connected to a tunnel that enters the ingress and exits the egress, i.e., the packet carried over the tunnel. This is sometimes known as the "tunneled packet", i.e., the packet carried over the tunnel.

- o Tunnel link packet: packets that traverse from ingress to egress, in which resides all or part of a tunnel transit packet. This is sometimes known as the "tunnel packet", i.e., the packet of the tunnel itself.

- o Link MTU (LMTU): the largest message that can transit a link. Note that this need not be the native size of messages on the link.

- o Reassembly MTU (RMTU): the largest message that can be reassembled by a receiver, and is not directly related to the link or path MTU. Sometimes also referred to as "receiver MTU".

- o Path MTU (PMTU): the largest message that can transit a path. Typically, this is the minimum of the link MTUs of the links of the path.

- o Tunnel MTU (TMTU): the largest message that can transit a tunnel. Typically, this is limited by the egress reassembly MTU.

3. The Tunnel Model

A network architecture is an abstract description of a distributed communications system, its components and their relationships, the requisite properties of those components and the emergent properties of the system that result [To03]. Such descriptions can help explain behavior, as when the OSI seven-layer model is used as a teaching example [Zi80]. Architectures describe capabilities - and, just as importantly, constraints.

A network can be defined as a system of endpoints and relays interconnected by communication paths, abstracting away issues of naming in order to focus on message forwarding. To the extent that the Internet has a single, coherent interpretation, its architecture is defined by its core protocols (IP [RFC791], TCP [RFC793], UDP [RFC768]) and messages, hosts, routers, and links [Cl88][To03], as shown in Figure 3:

```
+------+    ------      ------    +------+
|      |   /      \    /      \   |      |
| HOST |--+ ROUTER +--+ ROUTER +--| HOST |
|      |   \      /    \      /   |      |
+------+    ------      ------    +------+
```

Figure 3 Basic Internet architecture

As a network architecture, the Internet is a system of hosts and routers interconnected by links that exchange messages when possible. "When possible" defines the Internet's "best effort" principle. The limited role of routers and links represents the End-to-End Principle [Sa84] and longest-prefix match enables hierarchical forwarding.

Although the definitions of host, router, and link seem absolute, they are often relative as viewed within the context of one OSI layer, each of which can be considered a distinct network architecture. An Internet gateway is a Layer 3 router when it transits IP datagrams but it acts as a Layer 2 host as it sources or sinks Layer 2 messages on attached links to accomplish this transit capability. In this way, a single device (Internet gateway) behaves as different components (router, host) at different layers.

Even though a single device may have multiple roles - even concurrently - at a given layer, each role is typically static and location-independent. An Internet gateway always acts as a Layer 2 host and that behavior does not depend on where the gateway is viewed from within Layer 2. In the context of a single layer, a device's behavior is modeled as a single component from all viewpoints in that layer.

3.1. What is a tunnel?

A tunnel can be modeled as a link in another network [To98][To01][To03]. In Figure 4, a source host (Hsrc) and destination host (Hdst) communicating over a network M in which two routers (Ra and Rd) are connected by a tunnel.

```
                --_                                  --
   +------+   /  \                               /  \   +------+
   | Hsrc |--+ Ra +----      --      --      ----+ Rd +--| Hdst |
   +------+   \  /    /\    /  \    /  \    /\    \  /   +------+
                --    /I \--+ Rb +--+ Rc +--/E \    --
                      \  /   \  /    \  /   \  /
                       \/     --      --     \/
                      <------ Network N ------->
   <------------------------ Network M ------------------------->
```

Figure 4 The big picture

The tunnel consists of two elements (ingress I, egress E), that lie along a path connected by a (possibly different) network N. Regardless of how the ingress and egress are connected, the tunnel serves as a link to the devices it connects (here, Ra and Rb).

IP packets arriving at the ingress are encapsulated to traverse network N. We call these packets "tunnel transit packets" because they will now transit the tunnel inside one or more "tunnel link packets". Tunnel link packets use the source address of the ingress and the destination address of the egress - using whatever address is appropriate to the Layer at which the ingress and egress operate (Layer 2, Layer 3, Layer 4, etc.). The egress decapsulates those messages, which then continue on network M as if emerging from a link. To tunnel transit packets, and to the routers the tunnel connects (Ra and Rb), the tunnel acts as a link.

The model of each component (ingress, egress) and the entire system (tunnel) depends on the layer from which you view the tunnel. From the perspective of the outermost hosts (Hsrc and Hdst), the tunnel appears as a link between two routers (Ra and Rd). For routers along the tunnel (e.g., Rb and Rc), the ingress and egress appear as the endpoint hosts and Hsrc and Hdst are invisible.

When the tunnel network (N) is implemented using the same protocol as the endpoint network (M), the picture looks flatter (Figure 5), as if it were running over a single network. However, note that this appearance is incorrect - nothing has changed. From the perspective of the endpoints, Rb and Rc and network N don’t exist and aren’t visible, and from the perspective of the tunnel, network M doesn’t exist. The fact that network N and M use the same protocol, and may traverse the same links is irrelevant.

```
                --_           --      --            --
     +------+   /  \    /\    /  \    /  \    /\    /  \   +------+
     | Hsrc |--+ Ra +--/I \--+ Rb +--+ Rc +--/E \--+ Rd +--| Hdst |
     +------+   \  /   \  /   \  /    \  /   \  /   \  /   +------+
                 --     \/     --      --     \/     --
                       <------ Network N ------->
     <------------------------ Network M ------------------------->
```

Figure 5 IP in IP network picture

3.2. View from the Outside

From outside the tunnel, to network M, the entire tunnel acts as a link (Figure 6). It may be numbered or unnumbered and the addresses associated with the ingress and egress are irrelevant from outside.

```
                --_                                  --
     +------+   /  \                                /  \   +------+
     | Hsrc |--+ Ra +------------------------------+ Rd +--| Hdst |
     +------+   \  /                                \  /   +------+
                 --                                   --
     <------------------------ Network M ------------------------->
```

Figure 6 Tunnels as viewed from the outside

A tunnel is effectively invisible to the network in which it resides, except that it behaves exactly as a link. Consequently [RFC3819] requirements for links supporting IP also apply to tunnels.

E.g., the IP datagram hop count (IPv4 Time-to-Live [RFC791] and IPv6 Hop Limit [RFC2460]) are decremented when traversing a router, not by traversing a link - or thus a tunnel. Tunnels have a tunnel MTU - the largest datagram that can transit, just as links have a corresponding link MTU. A link MTU may not reflect the native link message sizes (ATM AAL5 48 byte messages support a 9KB MTU) and the same is true for a tunnel.

3.3. View from the Inside

Within network N, i.e., from inside the tunnel itself, the ingress is a source of tunnel link packets and the egress is a sink - both are hosts on network N (Figure 7). Consequently [RFC1122] Internet host requirements apply to ingress and egress nodes when Network N uses IP (and thus the ingress/egress use IP encapsulation).

```
                  _          --      --
                      /\    /  \    /  \    /\
                     /I \--+ Rb +--+ Rc +--/E \
                     \  /   \  /    \  /   \  /
                      \/     --      --     \/
                     <------ Network N ------->
```

Figure 7 Tunnels, as viewed from within the tunnel

Viewed from within the tunnel, the outer network (M) doesn’t exist. Tunnel link packets can be fragmented by the source (ingress) and reassembled at the destination (egress), just as at any endpoint. The path between ingress and egress may have a path MTU but the endpoints can exchange messages as large as can be reassembled at the destination (egress), i.e., an egress MTU. Information about the network - i.e., regarding MTU sizes, network reachability, etc. - are relayed from the destination (egress) and intermediate routers back to the source (ingress), without regard for the external network (M).

3.4. Location of the Ingress and Egress

The ingress and egress are endpoints of the tunnel and the tunnel is a link. The ingress and egress are thus link endpoints at the network nodes the tunnel interconnects. Such link endpoints are typically described as "network interfaces".

Tunnel interfaces may be physical or virtual. The interface may be implemented inside the node where the tunnel attaches, e.g., inside a host or router. The interface may also be implemented as a "bump in the wire" (BITW), somewhere along a link between the two nodes the link interconnects. IP in IP tunnels are often implemented as interfaces, where IPsec tunnels are sometimes implemented as BITW. These implementation variations determine only whether information available at the link endpoints (ingress/egress) can be easily shared with the connected network nodes.

3.5. Implications of This Model

This approach highlights a few key features of a tunnel as a network architecture construct:

- To the tunnel transit packets, tunnels turn a network (Layer 3) path into a (Layer 2) link

- To devices the tunnel traverses, the tunnel ingress and egress act as hosts that source and sink tunnel link packets

The consequences of these features are as follow:

- o Like a link, a tunnel has an MTU defined by the reassembly MTU of the receiving interface (egress).

- o Path MTU discovery in the network layer (i.e., outer network M) has no direct relation to the MTU of the hops within the link layer of the links (or thus tunnels) that connect its components.

- o Hops remain defined as the number of routers encountered on a path or the time spent at a router [RFC1812]. Hops are not decremented solely by the transit of a link, e.g., a packet with a hop count of zero should successfully transit a link (and thus a tunnel) that connects two hosts.

- o The addresses of a tunnel ingress and egress correspond to link layer addresses to the tunnel transit packet and outer network M. Many point-to-point tunnels are unnumbered in the network in which they reside (even though they must have addresses in the network they transit).

- o Like network interfaces, the ingress and egress are never a direct source of ICMP messages but may provide information to their attached host or router to generate those ICMP messages.

These observations make it much easier to determine what a tunnel must do to transit IP packets, notably it must satisfy all requirements expected of a link.

4. IP Tunnel Requirements

The requirements of an IP tunnel are defined by the requirements of an IP link because both transit IP packets. A tunnel must transit the IP MTU, i.e., 68B for IPv4 and 1280B for IPv6, and a tunnel must support address resolution when there is more than one egress.

The requirements of the tunnel ingress and egress are defined by the network over which they exchange messages (tunnel link packets). For IP-over-IP, this means that the ingress MUST NOT exceed the (fragment) Identification field uniqueness requirements [RFC6864].

These requirements remain even though tunnels have some unique issues, including the need for additional space for encapsulation headers and the potential for tunnel path MTU variation.

4.1. Fragmentation

As with any link layer, the MTU of a tunnel is defined as the receiving interface reassembly MTU, and must satisfy the requirements of the IP packets the tunnel transits.

Note that many of the issues with tunnel fragmentation and MTU handling were discussed in [RFC4459], but that document described a variety of alternatives as if they were independent. This document explains the combined approach that is necessary.

An IPv4 tunnel must transit 68 byte packets without further fragmentation [RFC791][RFC1122] and an IPv6 tunnel must transit 1280 byte packets without further fragmentation [RFC2460]. The tunnel MTU interacts with routers or hosts it connects the same way as would a link MTU. In the following pseudocode, TTPsize is the size of the tunnel transit packet, and egressRMTU is the receive MTU of the egress. As with any link, the link MTU is defined not by the native path of the link (the path MTU inside the tunnel) but by the egress reassembly MTU (egressRMTU). This is because the ICMP "packet too big" message indicates failure, not preference. There is no ICMP message for "larger than I'd like, but I can still transit it".

These rules apply at the host/router where the tunnel is attached:

```
if (TTP > linkMTU) then
   if (TTP can be fragmented, e.g., IPv4 DF=0) then
      split TTP into fragments of TunMTU size
      and send each fragment into the tunnel ingress
   else
      drop TTP and send ICMP "too big" to TTP source
   endif
else
   send TTP into the tunnel "interface" (the ingress)
endif
```

These rules apply at the tunnel ingress:

```
if (sizeof(TTP) <= TunnelPathMTU) then
   encapsulate TTP as received and emit
else
   if (TunnelPathMTU < sizeof(TTP) <= egressRMTU) then
      fragment TTP into TunMTU chunks
      encapsulate and emit each TTP
   else
      {never happens; host/router already dropped by now}
   endif
endif
```

For IPv4 or IPv6 over IPv6, the tunnel path MTU is a minimum of 1280 minus the encapsulation header (40 bytes) with its options (TOptSz) and the egress reassembly MTU is 1500 minus the same amount:

```
if (sizeof(TTP) <= (1240 - TOptSz)) then
   encapsulate TTP as received and emit
else
   if ((1240 - TOptSz) < sizeof(TTP) <= (1460 - TOptSz)) then
      fragment TTP into (1240 - TOptSz) chunks
      encapsulate and emit each TTP
   else
      {never happens; host/router already dropped by now}
   endif
endif
```

This tunnel supports IPv6 transit only if TOptSize is smaller than 180 bytes, and supports IPv4 transit if TOptSize is smaller than 884 bytes. IPv6 tunnel transit packets of 1280 bytes may be guaranteed transit the outer network (M) without needing fragmentation there but they may require ongoing fragmentation and reassembly if the tunnel MTU is not at least 1320 bytes.

When using IP directly over IP, the minimum egress reassembly MTU for IPv4 is 576 bytes and for IPv6 is 1500 bytes. This means that tunnels of IPv4-over-IPv4, IPv4-over-IPv6, and IPv6-over-IPv6 are possible without additional requirements, but this may involve ingress fragmentation and egress reassembly. IPv6 cannot be tunneled directly over IPv4 without additional requirements, notably that the egress reassembly MTU or the link path MTU are at least 1280 bytes. Fragmentation and reassembly cannot be avoided for IPv6-over-IPv6 without similar requirements.

When ongoing ingress fragmentation and egress reassembly would be prohibitive or costly, larger MTUs can be supported by design and confirmed either out-of-band (by design) or in-band (e.g., using PLMTUD [RFC4821], as done in SEAL [RFC5320] and AERO [Te15]). Alternately, an ingress can encapsulate packets that fit and shut down once fragmentation is needed, but it must not continue to forward smaller packets while dropping larger packets that are still within required limits.

4.2. MTU discovery

MTU discovery enables a network path to support a larger path MTU and egress MTU than it can assume from the protocol over which it operates. There are two ways in which MTU discovery interact with tunnels: the MTU of the path over the tunnel and the MTU of the tunnel itself.

A tunnel has two different MTU values: the largest payload that can traverse from ingress to egress without further fragmentation (the tunnel path MTU) and the largest payload that can traverse from ingress to egress. The latter is defined by the egress reassembly MTU, not the tunnel path MTU, and is the tunnel MTU.

The path MTU over the tunnel is limited by the tunnel MTU (the egress reassembly MTU) but not the tunnel path MTU. There is temptation to optimize tunnel traversal so that packets are not fragmented between ingress and egress, i.e., to tune the network path MTU to the tunnel link MTU. This is hazardous for many reasons:

- The tunnel is capable of transiting packets as large as the egress reassembly MTU, which is always at least as large as the tunnel path MTU and typically is larger.

- ICMP has only one type of error message regarding large packets - "too big", i.e., too large to transit. There is no optimization message of "bigger than I’d like, but I can deal with if needed".

- IP tunnels often involve some level of recursion, i.e., encapsulation over itself [RFC4459].

Recursive tunneling occurs whenever a protocol ends up encapsulated in itself. This happens directly, as when IPv4 is encapsulated in IPv4, or indirectly, as when IP is encapsulated in UDP which then is a payload inside IP. It can involve many layers of encapsulation because a tunnel provider isn’t always aware of whether the packets it transits are already tunneled.

Recursion is impossible when the tunnel transit packets are limited to that of the native size of the tunnel path MTU. Arriving tunnel transit packets have a minimum supported size (1280 for IPv6) and the tunnel path MTU has the same size; there would be no room for the additional encapsulation headers. The result would be an IPv6 tunnel that cannot satisfy IPv6 transit requirements.

It is more appropriate to require the tunnel to satisfy IP transit requirements and enforce that requirement at design time or during operation (the latter using PLMTUD [RFC4821]). Conventional path MTU discovery (PMTUD) relies existing endpoint ICMP processing of explicit negative feedback from routers along the path via "message to big" ICMP packets in the reverse direction of the tunnel [RFC1191]. This technique is susceptible to the "black hole" phenomenon, in which the ICMP messages never return to the source due to policy-based filtering [RFC2923]. PLMTUD requires a separate, direct control channel from the egress to the ingress that provides positive feedback; the direct channel is not blocked by policy filters and the positive feedback ensures fail-safe operation if feedback messages are lost [RFC4821].

4.3. IP ID exhaustion

In IPv4, the IP Identification (ID) field is a 16-bit value that is unique for every packet for a given source address, destination address, and protocol, such that it does not repeat within the Maximum Segment Lifetime (MSL) [RFC791][RFC1122]. Although the ID field was originally intended for fragmentation and reassembly, it can also be used to detect and discard duplicate packets, e.g., at congested routers (see Sec. 3.2.1.5 of [RFC1122]). For this reason, and because IPv4 packets can be fragmented anywhere along a path, all packets between a source and destination of a given protocol must have unique ID values over a period of an MSL, which is typically interpreted as two minutes (120 seconds). These requirements have recently been somewhat relaxed in recognition of the primary use of this field for reassembly and the need to handle only fragment misordering at the receiver [RFC6864].

The uniqueness of the IP ID is a known problem for high speed devices, because it limits the speed of a single protocol between two endpoints [RFC4963]. Although this suggests that the uniqueness of the IP ID is moot, tunnels exacerbate this condition. A tunnel often aggregates traffic from a number of different source and destination addresses, of different protocols, and encapsulates them in a header with the same ingress and egress addresses, all using a single encapsulation protocol. The result is one of the following:

1. The IP ID rules are enforced, and the tunnel throughput is severely limited.

2. The IP ID rules are enforced, and the tunnel consumes large numbers of ingress/egress IP addresses solely to ensure ID uniqueness.

3. The IP ID rules are ignored.

The last case is the most obvious solution, because it corresponds to how endpoints currently behave. Fortunately, fragmentation is somewhat rare in the current Internet at large, but it can be common along a tunnel. Fragments that repeat the IP ID risk being reassembled incorrectly, especially when fragments are reordered or lost. Reassembly errors are not always detected by other protocol layers (see Sec. 4.8), and even when detected they can result in excessive overall packet loss and can waste bandwidth between the egress and ultimate packet destination.

4.4. Hop Count

This section considers the selection of the value of the hop count of the tunnel link header, as well as the potential impact on the tunnel transit header. The former is affected by the number of hops within the tunnel. The latter determines whether the tunnel has visible effect on the transit packet.

In general, the Internet hop count field is used to detect and avoid forwarding loops that cannot be corrected without a synchronized reboot. The IPv4 Time-to-Live (TTL) and IPv6 Hop Limit field each serve this purpose [RFC791][RFC2460].

The IPv4 TTL field was originally intended to indicate packet expiration time, measured in seconds. A router is required to decrement the TTL by at least one or the number of seconds the packet is delayed, whichever is larger [RFC1812]. Packets are rarely held that long, and so the field has come to represent the count of the number of routers traversed. IPv6 makes this meaning more explicit.

These hop count fields represent the number of network forwarding elements traversed by an IP datagram. An IP datagram with a hop count of zero can traverse a link between two hosts because it never visits a router (where it would need to be decremented and would have been dropped).

An IP datagram traversing a tunnel thus need not have its hopcount modified, i.e., the tunnel transit header need not be affected. A

zero hop count datagram should be able to traverse a tunnel as easily as it traverses a link. A router MAY be configured to decrement packets traversing a particular link (and thus a tunnel), which may be useful in emulating a path as if it had traversed one or more routers, but this is strictly optional. The ability of the outer network and tunnel network to avoid indefinitely looping packets does not rely on the hop counts of the tunnel traversal packet and tunnel link packet being related in any way at all.

The hop count field is also used by several protocols to determine whether endpoints are "local", i.e., connected to the same subnet (link-local discovery and related protocols [RFC4861]). A tunnel is a way to make a remote address appear directly-connected, so it makes sense that the other ends of the tunnel appear local and that such link-local protocols operate over tunnels unless configured explicitly otherwise. When the interfaces of a tunnel are numbered, these can be interpreted the same way as if they were on the same link subnet.

4.5. Signaling

In the current Internet architecture, signaling goes upstream, either from routers along a path or from the destination, back toward the source. Such signals are typically contained in ICMP messages, but can involve other protocols such as RSVP, transport protocol signals (e.g., TCP RSTs), or multicast control or transport protocols.

A tunnel behaves like a link and acts like a link interface at the nodes where it is attached. As such, it can provide information that enhances IP signaling (e.g., ICMP), but itself does not directly generate ICMP messages.

For tunnels, this means that there are two separate signaling paths. The outer network M devices can each signal the source of the tunnel transit packets, Hsrc (Figure 8). Inside the tunnel, the inner network N devices can signal the source of the tunnel link packets, the ingress I (Figure 9).

```
        +--------+-----------------------------------+--------+
        |        |                                   |        |
        v        --_                                 --       v
     +------+   /  \                               /  \   +------+
     | Hsrc |--+ Ra +----      --      --      ----+ Rd +--| Hdst |
     +------+   \  /    /\    /  \    /  \    /\    \  /   +------+
                --    /I \--+ Rb +--+ Rc +--/E \    --
                      \  /   \  /    \  /   \  /
                       \/     --      --     \/
                      <------ Network N ------->
     <------------------------ Network M ------------------------->
```

Figure 8 Signals outside the tunnel

```
                         +-----+-------+------+
                --_      |     |       |      |      --
     +------+   /  \     v     |       |      |     /  \   +------+
     | Hsrc |--+ Ra +----      --      --      ----+ Rd +--| Hdst |
     +------+   \  /    /\    /  \    /  \    /\    \  /   +------+
                --    /I \--+ Rb +--+ Rc +--/E \    --
                      \  /   \  /    \  /   \  /
                       \/     --      --     \/
                      <------ Network N ------->
     <------------------------ Network M ------------------------->
```

Figure 9 Signals inside the tunnel

These two signal paths are inherently distinct except where information is exchanged between the network interface of the tunnel (the ingress) and its attached device (Ra, in both figures).

It is always possible for a network interface to provide hints to its attached device (host or router), which can be used for optimization. In this case, when signals inside the tunnel indicate a change to the tunnel, the ingress (i.e., the tunnel network interface) can provide information to the router (Ra, in both figures), so that Ra can generate the appropriate signal in return to Hsrc. This relaying may be difficult, because signals inside the tunnel may not return enough information to the ingress to support direct relaying to Hsrc.

In all cases, the tunnel ingress needs to determine how to relay the signals from inside the tunnel into signals back to the source. For some protocols this is either simple or impossible (such as for ICMP), for others, it can even be undefined (e.g., multicast). In some cases, the individual signals relayed from inside the tunnel may result in corresponding signals in the outside network, and in other cases they may just change state of the tunnel interface. In the

latter case, the result may cause the router Ra to generate new ICMP errors when later messages arrive from Hsrc or other sources in the outer network.

The meaning of the relayed information must be carefully translated. In the case of soft or hard ICMP errors, the translation may be obvious. ICMP "packet too big" messages from inside the tunnel do not necessarily have a direct impact on Ra unless they arrive from the egress (where they would update egressRMTU). Inside the tunnel, these messages could be used to adjust the ingress fragmentation.

In addition to ICMP, messages typically considered for translation include Explicit Congestion Notification (ECN [RFC6040]) and multicast (IGMP, e.g.).

4.6. Relationship of Header Fields

Some tunnel specifications attempt to relate the fields of the tunnel transit packet and tunnel link packet, i.e., the packet arriving at the ingress and the encapsulation header. These two headers are effectively independent and there is no utility in requiring their contents to be related.

In specific, the encapsulation header source and destination addresses are network endpoints in the tunnel network N, but have no meaning in the outer network M, even when the tunneled packet traverses the same network. The addresses are effectively independent, and the tunnel endpoint addresses are link addresses to the tunnel transit packet.

Because the tunneled packet uses source and destination addresses with a separate meaning, it is inappropriate to copy or reuse the IPv4 Identification or IPv6 Fragment ID fields of the tunnel transit packet. These fields need to be generated based on the context of the encapsulation header, not the tunnel transit header.

Similarly, the DF field need not be copied from the tunnel transit packet to the encapsulation header of the tunnel link packet (presuming both are IPv4). Path MTU discovery inside the tunnel does not directly correspond to path MTU discovery outside the tunnel.

The same is true for most other fields. When a field value is generated in the encapsulation header, its meaning should be derived from what is desired in the context of the tunnel as a link. When feedback is received from these fields, they should be presented to the tunnel ingress and egress as if they were network interfaces. The

behavior of the node where these interfaces attach should be identical to that of a conventional link.

There are exceptions to this rule that are explicitly intended to relay signals from inside the tunnel to outside the tunnel. The primary example is ECN [RFC6040], which copies the ECN bits from the tunnel transit header to the tunnel link header during encapsulation at the ingress and modifies the tunnel transit header at egress based on a combination of the bits of the two headers. This is intended to allow congestion notification within the tunnel to be interpreted as if it were on the direct path. Other examples may involve the DSCP flags. In both cases, it is assumed that the intent of copying values on encapsulation and merging values on decapsulation has the effect of allowing the tunnel to act as if it participates in the same type of network as outside the tunnel (network M).

4.7. Congestion

In general, tunnels carrying IP traffic need not react directly to congestion any more than would any other link layer [RFC5405]. IP traffic is not generally expected to be congestion reactive.

[text from David Black on ECN relaying?]

4.8. Checksums

IP traffic transiting a tunnel needs to expect a similar level of error detection and correction as it would expect from any other link. In the case of IPv4, there are no such expectations, which is partly why it includes a header checksum [RFC791].

IPv6 omitted the header checksum because it already expects most link errors to be detected and dropped by the link layer and because it also assumes transport protection [RFC2460]. When transiting IPv6 over IPv6, the tunnel fails to provide the expected error detection. This is why IPv6 is often tunneled over layers that include separate protection, such as GRE [RFC2784].

The fragmentation created by the tunnel ingress can increase the need for stronger error detection and correction, especially at the tunnel egress to avoid reassembly errors. The Internet checksum is known to be susceptible to reassembly errors that could be common [RFC4963], and should not be relied upon for this purpose. This is why SEAL and AERO include a separate checksum [RFC5320][Te15]. This requirement can be undermined when using UDP as a tunnel with no UDP checksum (as per [RFC6935][RFC6936]) when fragmentation occurs because the egress has no checksum with which to validate reassembly. For this reason,

it is safe to use UDP with a zero checksum for atomic (non-fragmented, non-fragmentable) tunnel link packets only; when used on fragments, whether generated at the ingress or en-route inside the tunnel, omission of such a checksum can result in reassembly errors that can cause additional work (capacity, forwarding processing, receiver processing) downstream of the egress.

4.9. Numbering

Tunnel ingresses and egresses have addresses associated with the encapsulation protocol. These addresses are the source and destination (respectively) of the encapsulated packet while traversing the tunnel network.

Tunnels may or may not have addresses in the network whose traffic they transit (e.g., network M in Figure 4). In some cases, the tunnel is an unnumbered interface to a point-to-point virtual link. When the tunnel has multiple egresses, tunnel interfaces require separate addresses in network M.

To see the effect of tunnel interface addresses, consider traffic sourced at router Ra in Figure 4. Even before being encapsulated by the ingress, that traffic needs a source IP network address that belongs to the router. One option is to use an address associated with one of the other interfaces of the router [RFC1122]. Another option is to assign a number to the tunnel interface itself. Regardless of which address is used, the resulting IP packet is then encapsulated by the tunnel ingress using the ingress address as a separate operation.

4.10. Multicast

[To be addressed]

Note that PMTU for multicast is difficult. PIM carries an option that may help in the Population Count Extensions to PIM [RFC6807].

IMO, again, this is no different than any other multicast link.

4.11. NAT / Load Balancing

[To be addressed]

4.12. Recursive tunnels.

The rules described in this document already support tunnels over tunnels, sometimes known as "recursive" tunnels, in which IP is

transited over IP either directly or via intermediate encapsulation (IP-UDP-IP).

There are known hazards to recursive tunneling, notably that the independence of the tunnel transit header and tunnel link header hop counts can result in a tunneling loop. Such looping can be avoided when using direct encapsulation (IP in IP) by use of a header option to track the encapsulation count and to limit that count [RFC2473]. This looping cannot be avoided when other protocols are used for tunneling, e.g., IP in UDP in IP, because the encapsulation count may not be visible where the recursion occurs.

5. Observations (implications)

[Leave this as a shopping list for now]

5.1. Tunnel protocol designers

Account for egress MTU/path MTU differences.

Include a stronger checksum.

Ensure the egress MTU is always larger than the path MTU.

Ensure that the egress reassembly can keep up with line rate OR design PLMTUD into the tunneling protocol.

5.2. Tunnel implementers

Detect when the egress MTU is exceeded.

Detect when the egress MTU drops below the required minimum and shut down the tunnel if that happens - configuring the tunnel down and issuing a hard error may be the only way to detect this anomaly, and it’s sufficiently important that the tunnel SHOULD be disabled.

Do NOT decrement the TTL as part of being a tunnel. It’s always already OK for a router to decrement the TTL based on different next-hop routers, but TTL is a property of a router not a link.

5.3. Tunnel operators

Keep the difference between "enforced by operators" vs. "enforced by active protocol mechanism" in mind. It’s fine to assume something the tunnel cannot or does not test, as long as you KNOW you can assume it. When the assumption is wrong, it will NOT be signaled by the tunnel. Do NOT decrement the TTL as part of being a tunnel. It’s

always already OK for a router to decrement the TTL based on different next-hop routers, but TTL is a property of a router not a link.

Do NOT decrement the TTL as part of being a tunnel. It’s always already OK for a router to decrement the TTL based on different next-hop routers, but TTL is a property of a router not a link.

5.4. For existing standards

5.4.1. Generic UDP Encapsulation (GUE - IP in UDP in IP)

[He15a][He15b]

5.4.2. Generic Packet Tunneling in IPv6

[RFC2473]

Consistent with this doc:

Considers the endpoints of the tunnel as virtual interfaces.

Considers the tunnel a virtual link.

Requires source fragmentation at the ingress and reassembly at the egress.

Includes a recursion limit to prevent unlimited re-encapsulation.

Sets tunnel transit header hop limit independently.

Sends ICMPs back at the ingress based on the arriving tunnel transit packet and its relation to the tunnel MTU (though it uses the incorrect value of the tunnel MTU; see below).

Allows for ingress relaying of internal tunnel errors (but see below; it does not discuss retaining state about these).

Inconsistent with this doc:

Decrements the tunnel transit header by 1, i.e., incorrectly assuming that tunnel endpoints occur at routers only and that the tunnel, rather than the router, is responsible for this decrement.

This doc goes to pains to describe the decapsulation process as if it were distinct from conventional protocol processing by the receiver (when it should not be).

Copies traffic class from tunnel link to tunnel transit header (as one variant).

Treats the tunnel MTU as the tunnel path MTU, rather than the tunnel egress MTU.

Incorrectly fragments IPv4 DF=0 tunnel transit packets that arrive larger than the tunnel MTU at the IPv6 layer; the relationship between IPv4 and the tunnel is more complex (as noted in this doc).

Fails to retain state from the tunnel based on ingress receiving ICMP messages from inside the tunnel, e.g., such as might cause future tunnel transit packets arriving at the ingress to be discarded with an ICMP error response rather than allowing them to proceed into the tunnel.

5.4.3. Geneve (NVO3)

[RFC7364][Gr15]

Consistent with this doc:

Generation of the link header fields is not discussed and presumed independent of transit packet.

Inconsistent with this doc:

Tries to match transit to tunnel path MTU rather than egress MTU.

5.4.4. GRE (IP in GRE in IP)

IPv4 [RFC2784][RFC7588][Pi15]:

Consistent with this doc:

Does not address link header generation.

Non-default behavior allows fragmentation of link packet to match tunnel path MTU up to the limit of the egress MTU.

Default behavior sets link DF independently.

Shuts the tunnel down if the tunnel path MTU isn’t => 1280.

Inconsistent with this doc:

Based on tunnel path MTU, not egress MTU.

Claims that the tunnel (GRE) mechanism is responsible for generating ICMP error messages.

Default behavior fragments transit packet (where possible) based on tunnel path MTU (it should fragment based on egress MTU).

Default behavior does not support the minimum MTU of IPv6 when run over IPv6.

Non-default behavior allows copying DF for IPv4 in IPv4.

5.4.5. IP in IP / mobile IP

IPv4 [RFC2003][RFC5944]:

Consistent with this doc:

Generate link ID independently

Generate link DF independently when transit DF=0

Generate ECN/update ECN based on sharing info [RFC6040]

Set link TTL to transit to egress only (independently)

Do not decrement TTL on entry except when part of forwarding

Do not decrement TTL on exit except when part of forwarding

Options not copied, but used as a hint to desired services.

Generally treat tunnel as a link, e.g., for link-local.

Inconsistent with this doc

Set link DF when transit DF=1 (won’t work unless I-E runs PLMTUD)

Drop at egress if transit TTL=0 (wrong TTL for host-host tunnels)

Drop when transit source is router’s IP (prevents tun from router)

Drop when transit source matches egress (prevents tun to router)

Use tunnel ICMPs to generate upper ICMPs, copying context (ICMPs are now coming from inside a link!); these should be handled by setting errors as a "network interface" and letting the attached host/router figure out what to send.

Using tunnel MTU discovery to tune the transit packet to the tunnel path MTU rather than egress MTU.

IPv6 [RFC2473]:

Consistent with this doc:

Doesn’t discuss lots of header fields, but implies they’re set independently.

Sets link TTL independently.

Inconsistent with this doc:

Tunnel issues ICMP PTBs.

ICMP PTB issued if larger then 1280 - header, rather than egress reassembly MTU.

Fragments IPv6 over IPv6 fragments only if transit is <= 1280 (i.e., forces all tunnels to have a max MTU of 1280).

Fragments IPv4 over IPv6 fragments only if IPv4 DF=0 (misinterpreting the "can fragment the IPv4 packet" as permission to fragment at the IPv6 link header)

Considers encapsulation a forwarding operation and decrements the transit TTL.

5.4.6. IPsec tunnel mode (IP in IPsec in IP)

[RFC4301]

Consistent with this doc:

Most of the rules, except as noted below.

Inconsistent with this doc:

Writes its own header copying rules (Sec 5.1.2), rather than referring to existing standards.

Uses policy to set, clear, or copy DF (policy isn’t the issue)

Intertwines tunneling with forwarding rather than presenting the tunnel as a network interface; this can be corrected by using IPsec transport mode with an IP-in-IP tunnel [RFC3884].

5.4.7. L2TP

[RFC3931]

Consistent with this doc:

Does not address most link headers, which are thus independent.

Inconsistent with this doc:

Manages tunnel access based on tunnel path MTU, instead of egress MTU.

Refers to RFC2473 (IPv6 in IPv6), which is inconsistent with this doc as noted above.

5.4.8. L2VPN

[RFC4664]

5.4.9. L3VPN

[RFC4176]

5.4.10. LISP

[RFC6830]

5.4.11. MPLS

[RFC3031]

5.4.12. PWE

[RFC3985]

5.4.13. SEAL/AERO

[RFC5320][Te15]

5.4.14. TRILL

[RFC5556][RFC6325]

Consistent with this doc:

Puts IP in Ethernet, so most of the issues don’t come up.

Ethernet doesn’t have TTL or fragment.

Rbridge (trill) TTL header is independent of transit packet.

5.5. For future standards

Larger IPv4 MTU (2K? or just 2x path MTU?) for reassembly

Always include frag support for at least two frags; do NOT try to deprecate fragmentation.

Limit encapsulation option use/space.

Augment ICMP to have two separate messages: PTB vs P-bigger-than-optimal

Include MTU as part of BGP as a hint - SB

Hazards of multi-MTU draft-van-beijnum-multi-mtu-04

6. Security Considerations

Tunnels may introduce vulnerabilities or add to the potential for receiver overload and thus DOS attacks. These issues are primarily related to the fact that a tunnel is a link that traverses a network path and to fragmentation and reassembly. ICMP signal translation introduces a new security issue and must be done with care. ICMP generation at the router or host attached to a tunnel is already covered by existing requirements (e.g., should be throttled).

Tunnels traverse multiple hops of a network path from ingress to egress. Traffic along such tunnels may be susceptible to on-path and off-path attacks, including fragment injection, reassembly buffer overload, and ICMP attacks. Some of these attacks may not be as visible to the endpoints of the architecture into which tunnels are deployed and these attacks may thus be more difficult to detect.

Fragmentation at routers or hosts attached to tunnels may place an undue burden on receivers where traffic is not sufficiently diffuse, because tunnels may induce source fragmentation at hosts and path fragmentation (for IPv4 DF=0) more for tunnels than for other links. Care should be taken to avoid this situation, notably by ensuring that tunnel MTUs are not significantly different from other link MTUs.

Tunnel ingresses emitting IP datagrams MUST obey all existing IP requirements, such as the uniqueness of the IP ID field. Failure to

either limit encapsulation traffic, or use additional ingress/egress IP addresses, can result in high speed traffic fragments being incorrectly reassembled.

[management?]

[Access control?]

describe relationship to [RFC6169] - JT (as per INTAREA meeting notes, don’t cover Teredo-specific issues in RFC6169, but include generic issues here)

7. IANA Considerations

This document has no IANA considerations.

The RFC Editor should remove this section prior to publication.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[Cl88] Clark, D., "The design philosophy of the DARPA internet protocols," Proc. Sigcomm 1988, p.106-114, 1988.

[Er94] Eriksson, H., "MBone: The Multicast Backbone," Communications of the ACM, Aug. 1994, pp.54-60.

[Gr15] Gross, J., et al., "Geneve: Generic Network Virtualization Encapsulation," draft-ietf-nvo3-geneve-00, May 2015.

[He15a] Herbert, T., L. Yong, O. Zia, "Generic UDP Encapsulation," draft-ietf-nvo3-gue-01, June 2015.

[He15b] Herbert, T., F. Templin, "Fragmentation option for Generic UDP Encapsulation," draft-herbert-gue-fragmentation-00, Mar. 2015.

[Pi15] Pignataro, C., R. Bonica, S. Krishnan, "IPv6 Support for Generic Routing Encapsulation (GRE)," draft-ietf-intarea-gre-ipv6-11, July 2015.

[RFC768] Postel, J, "User Datagram Protocol," RFC 768, Aug. 1980

[RFC791] Postel, J., "Internet Protocol," RFC 791 / STD 5, September 1981.

[RFC793] Postel, J, "Transmission Control Protocol," RFC 793, Sept. 1981.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers," RFC 1122 / STD 3, October 1989.

[RFC1191] Mogul, J., S. Deering, "Path MTU discovery," RFC 1191, November 1990.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers," RFC 1812, June 1995.

[RFC2003] Perkins, C., "IP Encapsulation within IP," RFC 2003, October 1996.

[RFC2460] Deering, S., R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, Dec. 1998.

[RFC2473] Conta, A., "Generic Packet Tunneling in IPv6 Specification," RFC 2473, Dec. 1998.

[RFC2784] Farinacci, D., T. Li, S. Hanks, D. Meyer, P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery," RFC 2923, September 2000.

[RFC2473] Conta, A., S. Deering, "Generic Packet Tunneling in IPv6 Specification," RFC 2473, Dec. 1998.

[RFC3031] Rosen, E., A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised" RFC 5944, Nov. 2010.

[RFC3819] Karn, P., Ed., C. Bormann, G. Fairhurst, D. Grossman, R. Ludwig, J. Mahdavi, G. Montenegro, J. Touch, L. Wood, "Advice for Internet Subnetwork Designers," RFC 3819 / BCP 89, July 2004.

[RFC3884] Touch, J., L. Eggert, Y. Wang, "Use of IPsec Transport Mode for Dynamic Routing," RFC 3884, September 2004.

[RFC3931] Lau, J., Ed., M. Townsley, Ed., I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)," RFC 3931, March 2005.

[RFC3985] Bryant, S., P. Pate (Eds.), "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

[RFC4176] El Mghazli, Y., Ed., T. Nadeau, M. Boucadair, K. Chan, A. Gonguet, "Framework for Layer 3 Virtual Private Networks (L3VPN) Operations and Management," RFC 4176, October 2005.

[RFC4301] Kent, S., and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, December 2005.

[RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling," RFC 4459, April 2006.

[RFC4664] Andersson, L., Ed., E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)," RFC 4664, September 2006.

[RFC4821] Mathis, M., J. Heffner, "Packetization Layer Path MTU Discovery," RFC 4821, March 2007.

[RFC4861] Narten, T., E. Nordmark, W. Simpson, H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," RFC 4861, Sept. 2007.

[RFC4963] Heffner, J., M. Mathis, B. Chandler, "IPv4 Reassembly Errors at High Data Rates," RFC 4963, July 2007.

[RFC5320] Templin, F., Ed., "The Subnetwork Encapsulation and Adaptation Layer (SEAL)," RFC 5320, Feb. 2010.

[RFC5405] Eggert, L., G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers," RFC 5405, Nov. 2008.

[RFC5556] Touch, J., R. Perlman, "Transparently Interconnecting Lots of Links (TRILL): Problem and Applicability Statement," RFC 5556, May 2009.

[RFC6040] Briscoe, B., "Tunneling of Explicit Congestion Notification," RFC 6040, Nov. 2010.

[RFC6169] Krishnan, S., D. Thaler, J. Hoagland, "Security Concerns With IP Tunneling," RFC 6169, Apr. 2011.

[RFC6325] Perlman, R., D. Eastlake, D. Dutt, S. Gai, A. Ghanwani, "Routing Bridges (RBridges): Base Protocol Specification," RFC 6325, July 2011.

[RFC6807] Farinacci, D., G. Shepherd, S. Venaas, Y. Cai, "Population Count Extensions to Protocol Independent Multicast (PIM)," RFC 6807, Dec. 2012.

[RFC6830] Farinacci, D., V. Fuller, D. Meyer, D. Lewis, "The Locator/ID Separation Protocol," RFC 6830, Jan. 2013.

[RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field," Proposed Standard, RFC 6864, Feb. 2013.

[RFC6935] Eubanks, M., P. Chimento, M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets," RFC 6935, Apr. 2013.

[RFC6936] Fairhurst, G., M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums," RFC 6936, Apr. 2013.

[RFC7364] Narten, T., Gray, E., Black, D., Fang, L., Kreeger, L., M. Napierala, "Problem Statement: Overlays for Network Virtualization", RFC 7364, October 2014.

[RFC7588] Bonica, R., C. Pignataro, J. Touch, "A Widely-Deployed Solution to the Generic Routing Encapsulation Fragmentation Problem," RFC 7588, July 2015.

[Sa84] Saltzer, J., D. Reed, D. Clark, "End-to-end arguments in system design," ACM Trans. on Computing Systems, Nov. 1984.

[Te15] Templin, F., "Asymmetric Extended Route Optimization," draft-templin-aerolink-58, June 2015.

[To01] Touch, J., "Dynamic Internet Overlay Deployment and Management Using the X-Bone," Computer Networks, July 2001, pp. 117-135.

[To03] Touch, J., Y. Wang, L. Eggert, G. Finn, "Virtual Internet Architecture," USC/ISI Tech. Report 570, Aug. 2003.

[To98] Touch, J., S. Hotz, "The X-Bone," Proc. Globecom Third Global Internet Mini-Conference, Nov. 1998.

[Zi80] Zimmermann, H., "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," IEEE Trans. on Comm., Apr. 1980.

9. Acknowledgments

This document originated as the result of numerous discussions among the authors, Jari Arkko, Stuart Bryant, Lars Eggert, Ted Faber, Gorry Fairhurst, Dino Farinacci, Matt Mathis, and Fred Templin, as well as members participating in the Internet Area Working Group.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292-6695
U.S.A.

Phone: +1 (310) 448-9151
Email: touch@isi.edu

W. Mark Townsley
Cisco
L'Atlantis, 11, Rue Camille Desmoulins
Issy Les Moulineaux, ILE DE FRANCE 92782

Email: townsley@cisco.com

Appendix A. Fragmentation

There are two places where fragmentation can occur in a tunnel, called Outer Fragmentation and Inner Fragmentation.

A.1. Outer Fragmentation

The simplest case is Outer Fragmentation, as shown in Figure 10. The bottom of the figure shows the network topology, where packets start at the source, enter the tunnel at the encapsulator, exit the tunnel at the decapsulator, and arrive finally at the destination. The packet traffic is shown above the topology, where the end-to-end packets are shown at the top. The packets are composed of an inner header (iH) and inner data (iD); the term "inner") is relative to the tunnel, as will become apparent. When the packet (iH,iD) arrives at the encapsulator, it is placed inside the tunnel packet structure, here shown as adding just an outer header, oH, in step (a).

When the encapsulated packet exceeds the MTU of the tunnel, the packet needs to be fragmented. In this case we fragment the packet at the outer header, with the fragments shown as (b1) and (b2). Note that the outer header indicates fragmentation (as ’ and "),the inner header occurs only in the first fragment, and the inner data is broken across the two packets. These fragments are reassembled at the encapsulator in step (c), and the resulting packet is decapsulated and sent on to the destination.

```
    +----+----+                                            +----+----+
    | iH | iD |------+ -  -  -  -  -  -  -  -  -  -  +------>| iH | iD |
    +----+----+      |                               |       +----+----+
                     v                               |
              +----+----+----+               +----+----+----+
          (a) | oH | iH | iD |               | oH | iH | iD | (c)
              +----+----+----+               +----+----+----+
                     |                               ^
                     |       +----+----+-----+       |
                (b1) +----- >| oH'| iH | iD1 |-------+
                     |       +----+----+-----+       |
                     |                               |
                     |       +----+-----+            |
                (b2) +----- >| oH"| iD2 |------------+
                             +----+-----+
   +-----+         +---+                          +---+         +-----+
   |     |        /     \ ======================= /     \        |     |
   | Src |=======|  Enc  |=======================|  Dec  |=======| Dst |
   |     |        \     / ======================= \     /        |     |
   +-----+         +---+                          +---+         +-----+
```

Figure 10 Fragmentation of the outer packet

Outer fragmentation isolates Source and Destination from tunnel encapsulation duties. This can be considered a benefit in clean, layered network design, but also may result in complex decapsulator design, especially where tunnels aggregate large amounts of traffic, such as IP ID overload (see Sec. 4.3). Outer fragmentation is valid for any tunnel encapsulation protocol that supports fragmentation (e.g., IPv4 or IPv6), where the tunnel endpoints act as the host endpoints of that protocol.

Along the tunnel, the inner header is contained only in the first fragment, which can interfere with mechanisms that 'peek' into lower layer headers, e.g., as for ICMP, as discussed in Sec. 4.5.

A.2. Inner Fragmentation

Inner Fragmentation distributes the impact of tunneling across both the decapsulator and destination, and is shown in Figure 11. Again, the network topology is shown at the bottom of the figure, and the original packets show at the top. Packets arrive at the encapsulator, and are fragmented there based on the inner header into (a1) and (a2). The fragments arrive at the decapsulator, which removes the outer header and forwards the resulting fragments on to the destination. The destination is then responsible for reassembling the fragments into the original packet.

```
+----+----+                                              +----+----+
| iH | iD |-------+-  -  -  -  -  -  -  -  -  -  -  -  - >| iH | iD |
+----+----+       |                                      +----+----+
                  v                                           ^
             +----+-----+                   +----+-----+      |
        (a1) | iH'| iD1 |                   | iH'| iD1 |------+
             +----+-----+                   +----+-----+      |
                                                              |
             +----+---                      +----+-----+      |
        (a2) | iH"| iD2 |                   | iH"| iD2 |------+
             +----+-----+                   +----+-----+
                  |                               ^
                  |       +----+----+-----        |
             (b1) +----- >| oH | iH'| iD1 |-------+
                  |       +----+----+-----+       |
                  |                               |
                  |       +----+----+-----+       |
             (b2) +----- >| oH | iH"| iD2 |-------+
                          +----+----+-----+
+-----+         +---+                          +---+         +-----+
|     |        /     \ ======================= /     \        |     |
| Src |=======|  Enc  |=======================|  Dec  |=======| Dst |
|     |        \     / ======================= \     /        |     |
+-----+         +---+                          +---+         +-----+
```

Figure 11 Fragmentation of the inner packet

As noted, inner fragmentation distributes the effort of tunneling across the decapsulator and destinations; this can be especially important when the tunnel aggregates large amounts of traffic. Note that this mechanism is thus valid only when the original source packets can be fragmented on-path, e.g., as in IPv4.

Along the tunnel, the inner headers are copied into each fragment, and so are available to mechanisms that 'peek' into headers (e.g., ICMP, as discussed in Sec. 4.5). Because fragmentation happens on the inner header, the impact of IP ID is reduced.

APPENDIX B: Fragmentation efficiency

B.1. Selecting fragment sizes

There are different ways to fragment a packet. Consider a network with an MTU as shown in Figure 12, where packets are encapsulated over the same network layer as they arrive on (e.g., IP in IP). If a packet as large as the MTU arrives, it must be fragmented to accommodate the additional header.

```
                X===========================X (MTU)
                +----+----------------------+
                | iH | DDDDDDDDDDDDDDDDDDDD |
                +----+----------------------+
                  |
                  |  X===========================X (MTU)
                  |  +---+----+------------------+
            (a) +->| H'| iH | DDDDDDDDDDDDDDDD |
                  |  +---+----+------------------+
                  |      |
                  |      |  X===========================X (MTU)
                  |      |  +----+---+----+-------------+
                  | (a1) +->| nH'| H | iH | DDDDDDDDDDD |
                  |      |  +----+---+----+-------------+
                  |      |
                  |      |  +----+-------+
                  | (a2) +->| nH"| DDDDD |
                  |         +----+-------+
                  |
                  |  +---+------+
            (b) +->| H"| DDDD |
                     +---+------+
                         |
                         |  +----+---+------+
                   (b1) +->| nH'| H"| DDDD |
                           +----+---+------+
```

Figure 12Fragmenting via maximum fit

Figure 12 shows this process, using Outer Fragmentation as an example (the situation is the same for Inner Fragmentation, but the headers that are affected differ). The arriving packet is first split into (a) and (b), where (a) is of the MTU of the network. However, this tunnel then traverses over another tunnel, whose impact the first tunnel ingress has not accommodated. The packet (a) arrives at the second tunnel ingress, and needs to be encapsulated again, but because it is already at the MTU, it needs to be fragmented as well,

into (a1) and (a2). In this case, packet (b) arrives at the second tunnel ingress and is encapsulated into (b1) without fragmentation, because it is already below the MTU size.

In Figure 13, the fragmentation is done evenly, i.e., by splitting the original packet into two roughly equal-sized components, (c) and (d). Note that (d) contains more packet data, because (c) includes the original packet header because this is an example of Outer Fragmentation. The packets (c) and (d) arrive at the second tunnel encapsulator, and are encapsulated again; this time, neither packet exceeds the MTU, and neither requires further fragmentation.

```
                X===========================X (MTU)
                +----+-----------------------+
                | iH | DDDDDDDDDDDDDDDDDDDD |
                +----+-----------------------+
                  |
                  |  X===========================X (MTU)
                  |  +---+----+----------+
            (c) +->| H'| iH | DDDDDDDD |
                  |  +---+----+----------+
                  |      |
                  |      |  X===========================X (MTU)
                  |      |  +----+---+----+----------+
                  | (c1) +->| nH | H'| iH | DDDDDDDD |
                  |         +----+---+----+----------+
                  |
                  |  +---+--------------+
            (d) +->| H"| DDDDDDDDDDDD |
                     +---+--------------+
                         |
                         |  +----+---+--------------+
                      (d1) +->| nH | H"| DDDDDDDDDDDD |
                               +----+---+--------------+
```

Figure 13 Fragmenting evenly

B.2. Packing

Encapsulating individual packets to traverse a tunnel can be inefficient, especially where headers are large relative to the packets being carried. In that case, it can be more efficient to encapsulate many small packets in a single, larger tunnel payload. This technique, similar to the effect of packet bursting in Gigabit Ethernet (regardless of whether they're encoded using L2 symbols as delineators), reduces the overhead of the encapsulation headers

(Figure 14). It reduces the work of header addition and removal at the tunnel endpoints, but increases other work involving the packing and unpacking of the component packets carried.

```
                  +-----+-----+
                  | iHa | iDa |
                  +-----+-----+
                        |
                        |     +-----+-----+
                        |     | iHb | iDb |
                        |     +-----+-----+
                        |           |
                        |           |     +-----+-----+
                        |           |     | iHc | iDc |
                        |           |     +-----+-----+
                        |           |           |
                        v           v           v
             +----+-----+-----+-----+-----+-----+-----+
             | oH | iHa | iHa | iHb | iDb | iHc | iDc |
             +----+-----+-----+-----+-----+-----+-----+
```

Figure 14 Packing packets into a tunnel

[NOTE: PPP chopping and coalescing?]

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

D. Black
EMC
J. Hudson
Brocade
L. Kreeger
Cisco
M. Lasserre
Alcatel-Lucent
T. Narten
IBM

October 19, 2015

An Architecture for Overlay Networks (NVO3)
draft-ietf-nvo3-arch-04

Abstract

This document presents a high-level overview architecture for building overlay networks in NVO3. The architecture is given at a high-level, showing the major components of an overall system. An important goal is to divide the space into individual smaller components that can be implemented independently and with clear interfaces and interactions with other components. It should be possible to build and implement individual components in isolation and have them work with other components with no changes to other components. That way implementers have flexibility in implementing individual components and can optimize and innovate within their respective components without requiring changes to other components.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document presents a high-level architecture for building overlay networks in NVO3. The architecture is given at a high-level, showing the major components of an overall system. An important goal is to divide the space into smaller individual components that can be implemented independently and with clear interfaces and interactions with other components. It should be possible to build and implement individual components in isolation and have them work with other components with no changes to other components. That way implementers have flexibility in implementing individual components and can optimize and innovate within their respective components without necessarily requiring changes to other components.

The motivation for overlay networks is given in [RFC7364]. "Framework for DC Network Virtualization" [RFC7365] provides a framework for discussing overlay networks generally and the various components that must work together in building such systems. This document differs from the framework document in that it doesn't attempt to cover all possible approaches within the general design space. Rather, it describes one particular approach.

2. Terminology

This document uses the same terminology as [RFC7365]. In addition, the following terms are used:

NV Domain A Network Virtualization Domain is an administrative construct that defines a Network Virtualization Authority (NVA), the set of Network Virtualization Edges (NVEs) associated with that NVA, and the set of virtual networks the NVA manages and supports. NVEs are associated with a (logically centralized) NVA, and an NVE supports communication for any of the virtual networks in the domain.

NV Region A region over which information about a set of virtual networks is shared. The degenerate case of a single NV Domain corresponds to an NV region corresponding to that domain. The more interesting case occurs when two or more NV Domains share information about part or all of a set of virtual networks that they manage. Two NVAs share information about particular virtual networks for the purpose of supporting connectivity between tenants located in different NV Domains. NVAs can share information about an entire NV domain, or just individual virtual networks.

Tenant System Identifier (TSI) Interface to a Virtual Network as presented to a Tenant System. The TSI logically connects to the NVE via a Virtual Access Point (VAP). To the Tenant System, the TSI is like a NIC; the TSI presents itself to a Tenant System as a normal network interface.

VLAN Unless stated otherwise, the terms VLAN and VLAN Tag are used in this document denote a C-VLAN [IEEE-802.1Q] and the terms are used interchangeably to improve readability.

3. Background

Overlay networks are an approach for providing network virtualization services to a set of Tenant Systems (TSs) [RFC7365]. With overlays, data traffic between tenants is tunneled across the underlying data center’s IP network. The use of tunnels provides a number of benefits by decoupling the network as viewed by tenants from the underlying physical network across which they communicate.

Tenant Systems connect to Virtual Networks (VNs), with each VN having associated attributes defining properties of the network, such as the set of members that connect to it. Tenant Systems connected to a virtual network typically communicate freely with other Tenant Systems on the same VN, but communication between Tenant Systems on

one VN and those external to the VN (whether on another VN or connected to the Internet) is carefully controlled and governed by policy.

A Network Virtualization Edge (NVE) [RFC7365] is the entity that implements the overlay functionality. An NVE resides at the boundary between a Tenant System and the overlay network as shown in Figure 1. An NVE creates and maintains local state about each Virtual Network for which it is providing service on behalf of a Tenant System.

```
        +--------+                                          +--------+
        | Tenant +--+                                  +----| Tenant |
        | System |  |                                 (')   | System |
        +--------+  |          ................       (   )  +--------+
                    |  +-+--+  .              .  +--+-+  (_)
                    |  | NVE|--.              .--| NVE|   |
                    +--|    |  .              .  |    |---+
                       +-+--+  .              .   +--+-+
                      /        .              .
                     /         .  L3 Overlay  .   +--+-++--------+
        +--------+  /          .    Network   .   | NVE|| Tenant |
        | Tenant +--+          .              .- -|    || System |
        | System |             .              .   +--+-++--------+
        +--------+             ................
                                      |
                                   +----+
                                   | NVE|
                                   |    |
                                   +----+
                                      |
                                      |
                          ====================
                              |               |
                          +--------+      +--------+
                          | Tenant |      | Tenant |
                          | System |      | System |
                          +--------+      +--------+
```

Figure 1: NVO3 Generic Reference Model

The following subsections describe key aspects of an overlay system in more detail. Section 3.1 describes the service model (Ethernet vs. IP) provided to Tenant Systems. Section 3.2 describes NVEs in more detail. Section 3.3 introduces the Network Virtualization Authority, from which NVEs obtain information about virtual networks.

Section 3.4 provides background on VM orchestration systems and their use of virtual networks.

3.1. VN Service (L2 and L3)

A Virtual Network provides either L2 or L3 service to connected tenants. For L2 service, VNs transport Ethernet frames, and a Tenant System is provided with a service that is analogous to being connected to a specific L2 C-VLAN. L2 broadcast frames are generally delivered to all (and multicast frames delivered to a subset of) the other Tenant Systems on the VN. To a Tenant System, it appears as if they are connected to a regular L2 Ethernet link. Within NVO3, tenant frames are tunneled to remote NVEs based on the MAC addresses of the frame headers as originated by the Tenant System. On the underlay, NVO3 packets are forwarded between NVEs based on the outer addresses of tunneled packets.

For L3 service, VNs transport IP datagrams, and a Tenant System is provided with a service that only supports IP traffic. Within NVO3, tenant frames are tunneled to remote NVEs based on the IP addresses of the packet originated by the Tenant System; any L2 destination addresses provided by Tenant Systems are effectively ignored. For L3 service, the Tenant System will be configured with an IP subnet that is effectively a point-to-point link, i.e., having only the Tenant System and a next-hop router address on it.

L2 service is intended for systems that need native L2 Ethernet service and the ability to run protocols directly over Ethernet (i.e., not based on IP). L3 service is intended for systems in which all the traffic can safely be assumed to be IP. It is important to note that whether NVO3 provides L2 or L3 service to a Tenant System, the Tenant System does not generally need to be aware of the distinction. In both cases, the virtual network presents itself to the Tenant System as an L2 Ethernet interface. An Ethernet interface is used in both cases simply as a widely supported interface type that essentially all Tenant Systems already support. Consequently, no special software is needed on Tenant Systems to use an L3 vs. an L2 overlay service.

NVO3 can also provide a combined L2 and L3 service to tenants. A combined service provides L2 service for intra-VN communication, but also provides L3 service for L3 traffic entering or leaving the VN. Architecturally, the handling of a combined L2/L3 service in NVO3 is intended to match what is commonly done today in non-overlay environments by devices providing a combined bridge/router service. With combined service, the virtual network itself retains the semantics of L2 service and all traffic is processed according to its

L2 semantics. In addition, however, traffic requiring IP processing is also processed at the IP level.

The IP processing for a combined service can be implemented on a standalone device attached to the virtual network (e.g., an IP router) or implemented locally on the NVE (see Section 5.4 on Distributed Gateways). For unicast traffic, NVE implementation of a combined service may result in a packet being delivered to another TS attached to the same NVE (on either the same or a different VN) or tunneled to a remote NVE, or even forwarded outside the NVO3 domain. For multicast or broadcast packets, the combination of NVE L2 and L3 processing may result in copies of the packet receiving both L2 and L3 treatments to realize delivery to all of the destinations involved. This distributed NVE implementation of IP routing results in the same network delivery behavior as if the L2 processing of the packet included delivery of the packet to an IP router attached to the L2 VN as a TS, with the router having additional network attachments to other networks, either virtual or not.

3.1.1. VLAN Tags in L2 Service

An NVO3 L2 virtual network service may include encapsulated L2 VLAN tags provided by a Tenant System, but does not use encapsulated tags in deciding where and how to forward traffic. Such VLAN tags can be passed through, so that Tenant Systems that send or expect to receive them can be supported as appropriate.

The processing of VLAN tags that an NVE receives from a TS is controlled by settings associated with the VAP. Just as in the case with ports on Ethernet switches, a number of settings could be imagined. For example, C-TAGs can be passed through transparently, they could always be stripped upon receipt from a Tenant System, they could be compared against a list of explicitly configured tags, etc.

Note that the handling of C-VIDs has additional complications, as described in Section 4.2.1 below.

3.1.2. TTL Considerations

For L3 service, Tenant Systems should expect the TTL of the packets they send to be decremented by at least 1. For L2 service, the TTL on packets (when the packet is IP) is not modified.

3.2. Network Virtualization Edge (NVE)

Tenant Systems connect to NVEs via a Tenant System Interface (TSI). The TSI logically connects to the NVE via a Virtual Access Point (VAP) and each VAP is associated with one Virtual Network as shown in

Figure 2. To the Tenant System, the TSI is like a NIC; the TSI presents itself to a Tenant System as a normal network interface. On the NVE side, a VAP is a logical network port (virtual or physical) into a specific virtual network. Note that two different Tenant Systems (and TSIs) attached to a common NVE can share a VAP (e.g., TS1 and TS2 in Figure 2) so long as they connect to the same Virtual Network.

```
                  |         Data Center Network (IP)        |
                  |                                         |
                  +-----------------------------------------+
                       |                           |
                       |       Tunnel Overlay      |
            +------------+---------+       +---------+------------+
            | +----------+-------+ |       | +-------+----------+ |
            | |  Overlay Module  | |       | |  Overlay Module  | |
            | +---------+--------+ |       | +---------+--------+ |
            |           |          |       |           |          |
      NVE1  |           |          |       |           |          | NVE2
            |  +--------+-------+  |       |  +--------+-------+  |
            |  | VNI1      VNI2 |  |       |  | VNI1      VNI2 |  |
            |  +-+----------+---+  |       |  +-+-----------+--+  |
            |    | VAP1     | VAP2 |       |    | VAP1      | VAP2|
            +----+----------+------+       +----+-----------+-----+
                 |          |                   |           |
                 |\         |                   |           |
                 | \        |                   |          /|
          -------+--\-------+-------------------+---------/-+-------
                 |   \      |     Tenant        |        /  |
            TSI1 |TSI2\     | TSI3            TSI1  TSI2/   TSI3
                +---+ +---+ +---+             +---+ +---+   +---+
                |TS1| |TS2| |TS3|             |TS4| |TS5|   |TS6|
                +---+ +---+ +---+             +---+ +---+   +---+
```

Figure 2: NVE Reference Model

The Overlay Module performs the actual encapsulation and decapsulation of tunneled packets. The NVE maintains state about the virtual networks it is a part of so that it can provide the Overlay Module with such information as the destination address of the NVE to tunnel a packet to, or the Context ID that should be placed in the encapsulation header to identify the virtual network that a tunneled packet belongs to.

On the data center network side, the NVE sends and receives native IP traffic. When ingressing traffic from a Tenant System, the NVE identifies the egress NVE to which the packet should be sent, adds an

overlay encapsulation header, and sends the packet on the underlay network. When receiving traffic from a remote NVE, an NVE strips off the encapsulation header, and delivers the (original) packet to the appropriate Tenant System. When the source and destination Tenant System are on the same NVE, no encapsulation is needed and the NVE forwards traffic directly.

Conceptually, the NVE is a single entity implementing the NVO3 functionality. In practice, there are a number of different implementation scenarios, as described in detail in Section 4.

3.3. Network Virtualization Authority (NVA)

Address dissemination refers to the process of learning, building and distributing the mapping/forwarding information that NVEs need in order to tunnel traffic to each other on behalf of communicating Tenant Systems. For example, in order to send traffic to a remote Tenant System, the sending NVE must know the destination NVE for that Tenant System.

One way to build and maintain mapping tables is to use learning, as 802.1 bridges do [IEEE-802.1Q]. When forwarding traffic to multicast or unknown unicast destinations, an NVE could simply flood traffic. While flooding works, it can lead to traffic hot spots and can lead to problems in larger networks.

Alternatively, to reduce the scope of where flooding must take place, or to eliminate it all together, NVEs can make use of a Network Virtualization Authority (NVA). An NVA is the entity that provides address mapping and other information to NVEs. NVEs interact with an NVA to obtain any required address mapping information they need in order to properly forward traffic on behalf of tenants. The term NVA refers to the overall system, without regards to its scope or how it is implemented. NVAs provide a service, and NVEs access that service via an NVE-to-NVA protocol as discussed in Section 4.3.

Even when an NVA is present, Ethernet bridge MAC address learning could be used as a fallback mechanism, should the NVA be unable to provide an answer or for other reasons. This document does not consider flooding approaches in detail, as there are a number of benefits in using an approach that depends on the presence of an NVA.

For the rest of this document, it is assumed that an NVA exists and will be used. NVAs are discussed in more detail in Section 7.

3.4. VM Orchestration Systems

VM orchestration systems manage server virtualization across a set of servers. Although VM management is a separate topic from network virtualization, the two areas are closely related. Managing the creation, placement, and movement of VMs also involves creating, attaching to and detaching from virtual networks. A number of existing VM orchestration systems have incorporated aspects of virtual network management into their systems.

Note also, that although this section uses the term "VM" and "hypervisor" throughout, the same issues apply to other virtualization approaches, including Linux Containers (LXC), BSD Jails, Network Service Appliances as discussed in Section 5.1, etc.. From an NVO3 perspective, it should be assumed that where the document uses the term "VM" and "hypervisor", the intention is that the discussion also applies to other systems, where, e.g., the host operating system plays the role of the hypervisor in supporting virtualization, and a container plays the equivalent role as a VM.

When a new VM image is started, the VM orchestration system determines where the VM should be placed, interacts with the hypervisor on the target server to load and start the VM and controls when a VM should be shutdown or migrated elsewhere. VM orchestration systems also have knowledge about how a VM should connect to a network, possibly including the name of the virtual network to which a VM is to connect. The VM orchestration system can pass such information to the hypervisor when a VM is instantiated. VM orchestration systems have significant (and sometimes global) knowledge over the domain they manage. They typically know on what servers a VM is running, and meta data associated with VM images can be useful from a network virtualization perspective. For example, the meta data may include the addresses (MAC and IP) the VMs will use and the name(s) of the virtual network(s) they connect to.

VM orchestration systems run a protocol with an agent running on the hypervisor of the servers they manage. That protocol can also carry information about what virtual network a VM is associated with. When the orchestrator instantiates a VM on a hypervisor, the hypervisor interacts with the NVE in order to attach the VM to the virtual networks it has access to. In general, the hypervisor will need to communicate significant VM state changes to the NVE. In the reverse direction, the NVE may need to communicate network connectivity information back to the hypervisor. Example VM orchestration systems in use today include VMware’s vCenter Server, Microsoft’s System Center Virtual Machine Manager, and systems based on OpenStack and its associated plugins (e.g., Nova and Neutron). Both can pass information about what virtual networks a VM connects to down to the

hypervisor. The protocol used between the VM orchestration system and hypervisors is generally proprietary.

It should be noted that VM orchestration systems may not have direct access to all networking related information a VM uses. For example, a VM may make use of additional IP or MAC addresses that the VM management system is not aware of.

4. Network Virtualization Edge (NVE)

As introduced in Section 3.2 an NVE is the entity that implements the overlay functionality. This section describes NVEs in more detail. An NVE will have two external interfaces:

Tenant System Facing: On the Tenant System facing side, an NVE interacts with the hypervisor (or equivalent entity) to provide the NVO3 service. An NVE will need to be notified when a Tenant System "attaches" to a virtual network (so it can validate the request and set up any state needed to send and receive traffic on behalf of the Tenant System on that VN). Likewise, an NVE will need to be informed when the Tenant System "detaches" from the virtual network so that it can reclaim state and resources appropriately.

Data Center Network Facing: On the data center network facing side, an NVE interfaces with the data center underlay network, sending and receiving tunneled TS packets to and from the underlay. The NVE may also run a control protocol with other entities on the network, such as the Network Virtualization Authority.

4.1. NVE Co-located With Server Hypervisor

When server virtualization is used, the entire NVE functionality will typically be implemented as part of the hypervisor and/or virtual switch on the server. In such cases, the Tenant System interacts with the hypervisor and the hypervisor interacts with the NVE. Because the interaction between the hypervisor and NVE is implemented entirely in software on the server, there is no "on-the-wire" protocol between Tenant Systems (or the hypervisor) and the NVE that needs to be standardized. While there may be APIs between the NVE and hypervisor to support necessary interaction, the details of such an API are not in-scope for the IETF to work on.

Implementing NVE functionality entirely on a server has the disadvantage that server CPU resources must be spent implementing the NVO3 functionality. Experimentation with overlay approaches and previous experience with TCP and checksum adapter offloads suggests that offloading certain NVE operations (e.g., encapsulation and

decapsulation operations) onto the physical network adapter can produce performance advantages. As has been done with checksum and/or TCP server offload and other optimization approaches, there may be benefits to offloading common operations onto adapters where possible. Just as important, the addition of an overlay header can disable existing adapter offload capabilities that are generally not prepared to handle the addition of a new header or other operations associated with an NVE.

While the exact details of how to split the implementation of specific NVE functionality between a server and its network adapters is an implementation matter and outside the scope of IETF standardization, the NVO3 architecture should be cognizant of and support such separation. Ideally, it may even be possible to bypass the hypervisor completely on critical data path operations so that packets between a TS and its VN can be sent and received without having the hypervisor involved in each individual packet operation.

4.2. Split-NVE

Another possible scenario leads to the need for a split NVE implementation. An NVE running on a server (e.g. within a hypervisor) could support NVO3 towards the tenant, but not perform all NVE functions (e.g., encapsulation) directly on the server; some of the actual NVO3 functionality could be implemented on (i.e., offloaded to) an adjacent switch to which the server is attached. While one could imagine a number of link types between a server and the NVE, one simple deployment scenario would involve a server and NVE separated by a simple L2 Ethernet link. A more complicated scenario would have the server and NVE separated by a bridged access network, such as when the NVE resides on a ToR, with an embedded switch residing between servers and the ToR.

For the split NVE case, protocols will be needed that allow the hypervisor and NVE to negotiate and setup the necessary state so that traffic sent across the access link between a server and the NVE can be associated with the correct virtual network instance. Specifically, on the access link, traffic belonging to a specific Tenant System would be tagged with a specific VLAN C-TAG that identifies which specific NVO3 virtual network instance it connects to. The hypervisor-NVE protocol would negotiate which VLAN C-TAG to use for a particular virtual network instance. More details of the protocol requirements for functionality between hypervisors and NVEs can be found in [I-D.ietf-nvo3-nve-nva-cp-req].

4.2.1. Tenant VLAN handling in Split-NVE Case

Preserving tenant VLAN tags across NVO3 as described in Section 3.1.1 poses additional complications in the split-NVE case. The portion of the NVE that performs the encapsulation function needs access to the specific VLAN tags that the Tenant System is using in order to include them in the encapsulated packet. When an NVE is implemented entirely within the hypervisor, the NVE has access to the complete original packet (including any VLAN tags) sent by the tenant. In the split-NVE case, however, the VLAN tag used between the hypervisor and offloaded portions of the NVE normally only identify the specific VN that traffic belongs to. In order to allow a tenant to preserve VLAN information in the split-NVE case, additional mechanisms would be needed.

4.3. NVE State

NVEs maintain internal data structures and state to support the sending and receiving of tenant traffic. An NVE may need some or all of the following information:

1. An NVE keeps track of which attached Tenant Systems are connected to which virtual networks. When a Tenant System attaches to a virtual network, the NVE will need to create or update local state for that virtual network. When the last Tenant System detaches from a given VN, the NVE can reclaim state associated with that VN.

2. For tenant unicast traffic, an NVE maintains a per-VN table of mappings from Tenant System (inner) addresses to remote NVE (outer) addresses.

3. For tenant multicast (or broadcast) traffic, an NVE maintains a per-VN table of mappings and other information on how to deliver tenant multicast (or broadcast) traffic. If the underlying network supports IP multicast, the NVE could use IP multicast to deliver tenant traffic. In such a case, the NVE would need to know what IP underlay multicast address to use for a given VN. Alternatively, if the underlying network does not support multicast, an NVE could use serial unicast to deliver traffic. In such a case, an NVE would need to know which remote NVEs are participating in the VN. An NVE could use both approaches, switching from one mode to the other depending on such factors as bandwidth efficiency and group membership sparseness.

4. An NVE maintains necessary information to encapsulate outgoing traffic, including what type of encapsulation and what value to use for a Context ID within the encapsulation header.

5. In order to deliver incoming encapsulated packets to the correct Tenant Systems, an NVE maintains the necessary information to map incoming traffic to the appropriate VAP (i.e., Tenant System Interface).

6. An NVE may find it convenient to maintain additional per-VN information such as QoS settings, Path MTU information, ACLs, etc.

4.4. Multi-Homing of NVEs

NVEs may be multi-homed. That is, an NVE may have more than one IP address associated with it on the underlay network. Multihoming happens in two different scenarios. First, an NVE may have multiple interfaces connecting it to the underlay. Each of those interfaces will typically have a different IP address, resulting in a specific Tenant Address (on a specific VN) being reachable through the same NVE but through more than one underlay IP address. Second, a specific tenant system may be reachable through more than one NVE, each having one or more underlay addresses. In both cases, NVE address mapping functionality needs to support one-to-many mappings and enable a sending NVE to (at a minimum) be able to fail over from one IP address to another, e.g., should a specific NVE underlay address become unreachable.

Finally, multi-homed NVEs introduce complexities when serial unicast is used to implement tenant multicast as described in Section 4.3. Specifically, an NVE should only receive one copy of a replicated packet.

Multi-homing is needed to support important use cases. First, a bare metal server may have multiple uplink connections to either the same or different NVEs. Having only a single physical path to an upstream NVE, or indeed, having all traffic flow through a single NVE would be considered unacceptable in highly-resilient deployment scenarios that seek to avoid single points of failure. Moreover, in today’s networks, the availability of multiple paths would require that they be usable in an active-active fashion (e.g., for load balancing).

4.5. VAP

The VAP is the NVE-side of the interface between the NVE and the TS. Traffic to and from the tenant flows through the VAP. If an NVE runs into difficulties sending traffic received on the VAP, it may need to signal such errors back to the VAP. Because the VAP is an emulation of a physical port, its ability to signal NVE errors is limited and lacks sufficient granularity to reflect all possible errors an NVE may encounter (e.g., inability reach a particular destination). Some

errors, such as an NVE losing all of its connections to the underlay, could be reflected back to the VAP by effectively disabling it. This state change would reflect itself on the TS as an interface going down, allowing the TS to implement interface error handling, e.g., failover, in the same manner as when a physical interfaces becomes disabled.

5. Tenant System Types

This section describes a number of special Tenant System types and how they fit into an NVO3 system.

5.1. Overlay-Aware Network Service Appliances

Some Network Service Appliances [I-D.ietf-nvo3-nve-nva-cp-req] (virtual or physical) provide tenant-aware services. That is, the specific service they provide depends on the identity of the tenant making use of the service. For example, firewalls are now becoming available that support multi-tenancy where a single firewall provides virtual firewall service on a per-tenant basis, using per-tenant configuration rules and maintaining per-tenant state. Such appliances will be aware of the VN an activity corresponds to while processing requests. Unlike server virtualization, which shields VMs from needing to know about multi-tenancy, a Network Service Appliance may explicitly support multi-tenancy. In such cases, the Network Service Appliance itself will be aware of network virtualization and either embed an NVE directly, or implement a split NVE as described in Section 4.2. Unlike server virtualization, however, the Network Service Appliance may not be running a hypervisor and the VM orchestration system may not interact with the Network Service Appliance. The NVE on such appliances will need to support a control plane to obtain the necessary information needed to fully participate in an NVO3 Domain.

5.2. Bare Metal Servers

Many data centers will continue to have at least some servers operating as non-virtualized (or "bare metal") machines running a traditional operating system and workload. In such systems, there will be no NVE functionality on the server, and the server will have no knowledge of NVO3 (including whether overlays are even in use). In such environments, the NVE functionality can reside on the first-hop physical switch. In such a case, the network administrator would (manually) configure the switch to enable the appropriate NVO3 functionality on the switch port connecting the server and associate that port with a specific virtual network. Such configuration would typically be static, since the server is not virtualized, and once

configured, is unlikely to change frequently. Consequently, this scenario does not require any protocol or standards work.

5.3. Gateways

Gateways on VNs relay traffic onto and off of a virtual network. Tenant Systems use gateways to reach destinations outside of the local VN. Gateways receive encapsulated traffic from one VN, remove the encapsulation header, and send the native packet out onto the data center network for delivery. Outside traffic enters a VN in a reverse manner.

Gateways can be either virtual (i.e., implemented as a VM) or physical (i.e., as a standalone physical device). For performance reasons, standalone hardware gateways may be desirable in some cases. Such gateways could consist of a simple switch forwarding traffic from a VN onto the local data center network, or could embed router functionality. On such gateways, network interfaces connecting to virtual networks will (at least conceptually) embed NVE (or split-NVE) functionality within them. As in the case with Network Service Appliances, gateways may not support a hypervisor and will need an appropriate control plane protocol to obtain the information needed to provide NVO3 service.

Gateways handle several different use cases. For example, one use case consists of systems supporting overlays together with systems that do not (e.g., bare metal servers). Gateways could be used to connect legacy systems supporting, e.g., L2 VLANs, to specific virtual networks, effectively making them part of the same virtual network. Gateways could also forward traffic between a virtual network and other hosts on the data center network or relay traffic between different VNs. Finally, gateways can provide external connectivity such as Internet or VPN access.

5.3.1. Gateway Taxonomy

As can be seen from the discussion above, there are several types of gateways that can exist in an NVO3 environment. This section breaks them down into the various types that could be supported. Note that each of the types below could be implemented in either a centralized manner or distributed to co-exist with the NVEs.

5.3.1.1. L2 Gateways (Bridging)

L2 Gateways act as layer 2 bridges to forward Ethernet frames based on the MAC addresses present in them.

L2 VN to Legacy L2: This type of gateway bridges traffic between L2 VNs and other legacy L2 networks such as VLANs or L2 VPNs.

L2 VN to L2 VN: The main motivation for this type of gateway to create separate groups of Tenant Systems using L2 VNs such that the gateway can enforce network policies between each L2 VN.

5.3.1.2. L3 Gateways (Only IP Packets)

L3 Gateways forward IP packets based on the IP addresses present in the packets.

L3 VN to Legacy L2: This type of gateway forwards packets on between L3 VNs and legacy L2 networks such as VLANs or L2 VPNs. The MAC address in any frames forwarded between the legacy L2 network would be the MAC address of the gateway.

L3 VN to Legacy L3: The type of gateway forwards packets between L3 VNs and legacy L3 networks. These legacy L3 networks could be local the data center, in the WAN, or an L3 VPN.

L3 VN to L2 VN: This type of gateway forwards packets on between L3 VNs and L2 VNs. The MAC address in any frames forwarded between the L2 VN would be the MAC address of the gateway.

L2 VN to L2 VN: This type of gateway acts similar to a traditional router that forwards between L2 interfaces. The MAC address in any frames forwarded between the L2 VNs would be the MAC address of the gateway.

L3 VN to L3 VN: The main motivation for this type of gateway to create separate groups of Tenant Systems using L3 VNs such that the gateway can enforce network policies between each L3 VN.

5.4. Distributed Inter-VN Gateways

The relaying of traffic from one VN to another deserves special consideration. Whether traffic is permitted to flow from one VN to another is a matter of policy, and would not (by default) be allowed unless explicitly enabled. In addition, NVAs are the logical place to maintain policy information about allowed inter-VN communication. Policy enforcement for inter-VN communication can be handled in (at least) two different ways. Explicit gateways could be the central point for such enforcement, with all inter-VN traffic forwarded to such gateways for processing. Alternatively, the NVA can provide such information directly to NVEs, by either providing a mapping for a target TS on another VN, or indicating that such communication is disallowed by policy.

When inter-VN gateways are centralized, traffic between TSs on different VNs can take suboptimal paths, i.e., triangular routing results in paths that always traverse the gateway. In the worst case, traffic between two TSs connected to the same NVE can be hair-pinned through an external gateway. As an optimization, individual NVEs can be part of a distributed gateway that performs such relaying, reducing or completely eliminating triangular routing. In a distributed gateway, each ingress NVE can perform such relaying activity directly, so long as it has access to the policy information needed to determine whether cross-VN communication is allowed. Having individual NVEs be part of a distributed gateway allows them to tunnel traffic directly to the destination NVE without the need to take suboptimal paths.

The NVO3 architecture must support distributed gateways for the case of inter-VN communication. Such support requires that NVO3 control protocols include mechanisms for the maintenance and distribution of policy information about what type of cross-VN communication is allowed so that NVEs acting as distributed gateways can tunnel traffic from one VN to another as appropriate.

Distributed gateways could also be used to distribute other traditional router services to individual NVEs. The NVO3 architecture does not preclude such implementations, but does not define or require them as they are outside the scope of NVO3.

5.5. ARP and Neighbor Discovery

For an L2 service, strictly speaking, special processing of ARP [RFC0826] (and IPv6 Neighbor Discovery (ND) [RFC4861]) is not required. ARP requests are broadcast, and NVO3 can deliver ARP requests to all members of a given L2 virtual network, just as it does for any packet sent to an L2 broadcast address. Similarly, ND requests are sent via IP multicast, which NVO3 can support by delivering via L2 multicast. However, as a performance optimization, an NVE can intercept ARP (or ND) requests from its attached TSs and respond to them directly using information in its mapping tables. Since an NVE will have mechanisms for determining the NVE address associated with a given TS, the NVE can leverage the same mechanisms to suppress sending ARP and ND requests for a given TS to other members of the VN. The NVO3 architecture must support such a capability.

6. NVE-NVE Interaction

Individual NVEs will interact with each other for the purposes of tunneling and delivering traffic to remote TSs. At a minimum, a control protocol may be needed for tunnel setup and maintenance. For

example, tunneled traffic may need to be encrypted or integrity protected, in which case it will be necessary to set up appropriate security associations between NVE peers. It may also be desirable to perform tunnel maintenance (e.g., continuity checks) on a tunnel in order to detect when a remote NVE becomes unreachable. Such generic tunnel setup and maintenance functions are not generally NVO3-specific. Hence, NVO3 expects to leverage existing tunnel maintenance protocols rather than defining new ones.

Some NVE-NVE interactions may be specific to NVO3 (and in particular be related to information kept in mapping tables) and agnostic to the specific tunnel type being used. For example, when tunneling traffic for TS-X to a remote NVE, it is possible that TS-X is not presently associated with the remote NVE. Normally, this should not happen, but there could be race conditions where the information an NVE has learned from the NVA is out-of-date relative to actual conditions. In such cases, the remote NVE could return an error or warning indication, allowing the sending NVE to attempt a recovery or otherwise attempt to mitigate the situation.

The NVE-NVE interaction could signal a range of indications, for example:

- "No such TS here", upon a receipt of a tunneled packet for an unknown TS.

- "TS-X not here, try the following NVE instead" (i.e., a redirect).

- Delivered to correct NVE, but could not deliver packet to TS-X (soft error).

- Delivered to correct NVE, but could not deliver packet to TS-X (hard error).

When an NVE receives information from a remote NVE that conflicts with the information it has in its own mapping tables, it should consult with the NVA to resolve those conflicts. In particular, it should confirm that the information it has is up-to-date, and it might indicate the error to the NVA, so as to nudge the NVA into following up (as appropriate). While it might make sense for an NVE to update its mapping table temporarily in response to an error from a remote NVE, any changes must be handled carefully as doing so can raise security considerations if the received information cannot be authenticated. That said, a sending NVE might still take steps to mitigate a problem, such as applying rate limiting to data traffic towards a particular NVE or TS.

7. Network Virtualization Authority

Before sending to and receiving traffic from a virtual network, an NVE must obtain the information needed to build its internal forwarding tables and state as listed in Section 4.3. An NVE can obtain such information from a Network Virtualization Authority.

The Network Virtualization Authority (NVA) is the entity that is expected to provide address mapping and other information to NVEs. NVEs can interact with an NVA to obtain any required information they need in order to properly forward traffic on behalf of tenants. The term NVA refers to the overall system, without regards to its scope or how it is implemented.

7.1. How an NVA Obtains Information

There are two primary ways in which an NVA can obtain the address dissemination information it manages. The NVA can obtain information either from the VM orchestration system, and/or directly from the NVEs themselves.

On virtualized systems, the NVA may be able to obtain the address mapping information associated with VMs from the VM orchestration system itself. If the VM orchestration system contains a master database for all the virtualization information, having the NVA obtain information directly to the orchestration system would be a natural approach. Indeed, the NVA could effectively be co-located with the VM orchestration system itself. In such systems, the VM orchestration system communicates with the NVE indirectly through the hypervisor.

However, as described in Section 4 not all NVEs are associated with hypervisors. In such cases, NVAs cannot leverage VM orchestration protocols to interact with an NVE and will instead need to peer directly with them. By peering directly with an NVE, NVAs can obtain information about the TSs connected to that NVE and can distribute information to the NVE about the VNs those TSs are associated with. For example, whenever a Tenant System attaches to an NVE, that NVE would notify the NVA that the TS is now associated with that NVE. Likewise when a TS detaches from an NVE, that NVE would inform the NVA. By communicating directly with NVEs, both the NVA and the NVE are able to maintain up-to-date information about all active tenants and the NVEs to which they are attached.

7.2. Internal NVA Architecture

For reliability and fault tolerance reasons, an NVA would be implemented in a distributed or replicated manner without single points of failure. How the NVA is implemented, however, is not important to an NVE so long as the NVA provides a consistent and well-defined interface to the NVE. For example, an NVA could be implemented via database techniques whereby a server stores address mapping information in a traditional (possibly replicated) database. Alternatively, an NVA could be implemented in a distributed fashion using an existing (or modified) routing protocol to maintain and distribute mappings. So long as there is a clear interface between the NVE and NVA, how an NVA is architected and implemented is not important to an NVE.

A number of architectural approaches could be used to implement NVAs themselves. NVAs manage address bindings and distribute them to where they need to go. One approach would be to use Border Gateway Protocol (BGP) [RFC4364] (possibly with extensions) and route reflectors. Another approach could use a transaction-based database model with replicated servers. Because the implementation details are local to an NVA, there is no need to pick exactly one solution technology, so long as the external interfaces to the NVEs (and remote NVAs) are sufficiently well defined to achieve interoperability.

7.3. NVA External Interface

Conceptually, from the perspective of an NVE, an NVA is a single entity. An NVE interacts with the NVA, and it is the NVA’s responsibility for ensuring that interactions between the NVE and NVA result in consistent behavior across the NVA and all other NVEs using the same NVA. Because an NVA is built from multiple internal components, an NVA will have to ensure that information flows to all internal NVA components appropriately.

One architectural question is how the NVA presents itself to the NVE. For example, an NVA could be required to provide access via a single IP address. If NVEs only have one IP address to interact with, it would be the responsibility of the NVA to handle NVA component failures, e.g., by using a "floating IP address" that migrates among NVA components to ensure that the NVA can always be reached via the one address. Having all NVA accesses through a single IP address, however, adds constraints to implementing robust failover, load balancing, etc.

In the NVO3 architecture, an NVA is accessed through one or more IP addresses (or IP address/port combination). If multiple IP addresses

are used, each IP address provides equivalent functionality, meaning that an NVE can use any of the provided addresses to interact with the NVA. Should one address stop working, an NVE is expected to failover to another. While the different addresses result in equivalent functionality, one address may respond more quickly than another, e.g., due to network conditions, load on the server, etc.

To provide some control over load balancing, NVA addresses may have an associated priority. Addresses are used in order of priority, with no explicit preference among NVA addresses having the same priority. To provide basic load-balancing among NVAs of equal priorities, NVEs could use some randomization input to select among equal-priority NVAs. Such a priority scheme facilitates failover and load balancing, for example, allowing a network operator to specify a set of primary and backup NVAs.

It may be desirable to have individual NVA addresses responsible for a subset of information about an NV Domain. In such a case, NVEs would use different NVA addresses for obtaining or updating information about particular VNs or TS bindings. A key question with such an approach is how information would be partitioned, and how an NVE could determine which address to use to get the information it needs.

Another possibility is to treat the information on which NVA addresses to use as cached (soft-state) information at the NVEs, so that any NVA address can be used to obtain any information, but NVEs are informed of preferences for which addresses to use for particular information on VNs or TS bindings. That preference information would be cached for future use to improve behavior - e.g., if all requests for a specific subset of VNs are forwarded to a specific NVA component, the NVE can optimize future requests within that subset by sending them directly to that NVA component via its address.

8. NVE-to-NVA Protocol

As outlined in Section 4.3, an NVE needs certain information in order to perform its functions. To obtain such information from an NVA, an NVE-to-NVA protocol is needed. The NVE-to-NVA protocol provides two functions. First it allows an NVE to obtain information about the location and status of other TSs with which it needs to communicate. Second, the NVE-to-NVA protocol provides a way for NVEs to provide updates to the NVA about the TSs attached to that NVE (e.g., when a TS attaches or detaches from the NVE), or about communication errors encountered when sending traffic to remote NVEs. For example, an NVE could indicate that a destination it is trying to reach at a destination NVE is unreachable for some reason.

While having a direct NVE-to-NVA protocol might seem straightforward, the existence of existing VM orchestration systems complicates the choices an NVE has for interacting with the NVA.

8.1. NVE-NVA Interaction Models

An NVE interacts with an NVA in at least two (quite different) ways:

- o NVEs embedded within the same server as the hypervisor can obtain necessary information entirely through the hypervisor-facing side of the NVE. Such an approach is a natural extension to existing VM orchestration systems supporting server virtualization because an existing protocol between the hypervisor and VM orchestration system already exists and can be leveraged to obtain any needed information. Specifically, VM orchestration systems used to create, terminate and migrate VMs already use well-defined (though typically proprietary) protocols to handle the interactions between the hypervisor and VM orchestration system. For such systems, it is a natural extension to leverage the existing orchestration protocol as a sort of proxy protocol for handling the interactions between an NVE and the NVA. Indeed, existing implementations can already do this.

- o Alternatively, an NVE can obtain needed information by interacting directly with an NVA via a protocol operating over the data center underlay network. Such an approach is needed to support NVEs that are not associated with systems performing server virtualization (e.g., as in the case of a standalone gateway) or where the NVE needs to communicate directly with the NVA for other reasons.

The NVO3 architecture will focus on support for the second model above. Existing virtualization environments are already using the first model. But they are not sufficient to cover the case of standalone gateways -- such gateways may not support virtualization and do not interface with existing VM orchestration systems.

8.2. Direct NVE-NVA Protocol

An NVE can interact directly with an NVA via an NVE-to-NVA protocol. Such a protocol can be either independent of the NVA internal protocol, or an extension of it. Using a purpose-specific protocol would provide architectural separation and independence between the NVE and NVA. The NVE and NVA interact in a well-defined way, and changes in the NVA (or NVE) do not need to impact each other. Using a dedicated protocol also ensures that both NVE and NVA implementations can evolve independently and without dependencies on each other. Such independence is important because the upgrade path for NVEs and NVAs is quite different. Upgrading all the NVEs at a

site will likely be more difficult in practice than upgrading NVAs because of their large number - one on each end device. In practice, it would be prudent to assume that once an NVE has been implemented and deployed, it may be challenging to get subsequent NVE extensions and changes implemented and deployed, whereas an NVA (and its associated internal protocols) are more likely to evolve over time as experience is gained from usage and upgrades will involve fewer nodes.

Requirements for a direct NVE-NVA protocol can be found in [I-D.ietf-nvo3-nve-nva-cp-req]

8.3. Propagating Information Between NVEs and NVAs

Information flows between NVEs and NVAs in both directions. The NVA maintains information about all VNs in the NV Domain, so that NVEs do not need to do so themselves. NVEs obtain from the NVA information about where a given remote TS destination resides. NVAs in turn obtain information from NVEs about the individual TSs attached to those NVEs.

While the NVA could push information about every virtual network to every NVE, such an approach scales poorly and is unnecessary. In practice, a given NVE will only need and want to know about VNs to which it is attached. Thus, an NVE should be able to subscribe to updates only for the virtual networks it is interested in receiving updates for. The NVO3 architecture supports a model where an NVE is not required to have full mapping tables for all virtual networks in an NV Domain.

Before sending unicast traffic to a remote TS (or TSes for broadcast or multicast traffic), an NVE must know where the remote TS(es) currently reside. When a TS attaches to a virtual network, the NVE obtains information about that VN from the NVA. The NVA can provide that information to the NVE at the time the TS attaches to the VN, either because the NVE requests the information when the attach operation occurs, or because the VM orchestration system has initiated the attach operation and provides associated mapping information to the NVE at the same time.

There are scenarios where an NVE may wish to query the NVA about individual mappings within an VN. For example, when sending traffic to a remote TS on a remote NVE, that TS may become unavailable (e.g,. because it has migrated elsewhere or has been shutdown, in which case the remote NVE may return an error indication). In such situations, the NVE may need to query the NVA to obtain updated mapping information for a specific TS, or verify that the information is still correct despite the error condition. Note that such a query

could also be used by the NVA as an indication that there may be an inconsistency in the network and that it should take steps to verify that the information it has about the current state and location of a specific TS is still correct.

For very large virtual networks, the amount of state an NVE needs to maintain for a given virtual network could be significant. Moreover, an NVE may only be communicating with a small subset of the TSs on such a virtual network. In such cases, the NVE may find it desirable to maintain state only for those destinations it is actively communicating with. In such scenarios, an NVE may not want to maintain full mapping information about all destinations on a VN. Should it then need to communicate with a destination for which it does not have mapping information, however, it will need to be able to query the NVA on demand for the missing information on a per-destination basis.

The NVO3 architecture will need to support a range of operations between the NVE and NVA. Requirements for those operations can be found in [I-D.ietf-nvo3-nve-nva-cp-req].

9. Federated NVAs

An NVA provides service to the set of NVEs in its NV Domain. Each NVA manages network virtualization information for the virtual networks within its NV Domain. An NV domain is administered by a single entity.

In some cases, it will be necessary to expand the scope of a specific VN or even an entire NV domain beyond a single NVA. For example, multiple data centers managed by the same administrator may wish to operate all of its data centers as a single NV region. Such cases are handled by having different NVAs peer with each other to exchange mapping information about specific VNs. NVAs operate in a federated manner with a set of NVAs operating as a loosely-coupled federation of individual NVAs. If a virtual network spans multiple NVAs (e.g., located at different data centers), and an NVE needs to deliver tenant traffic to an NVE that is part of a different NV Domain, it still interacts only with its NVA, even when obtaining mappings for NVEs associated with a different NV Domain.

Figure 3 shows a scenario where two separate NV Domains (1 and 2) share information about Virtual Network "1217". VM1 and VM2 both connect to the same Virtual Network 1217, even though the two VMs are in separate NV Domains. There are two cases to consider. In the first case, NV Domain B (NVB) does not allow NVE-A to tunnel traffic directly to NVE-B. There could be a number of reasons for this. For example, NV Domains 1 and 2 may not share a common address space

(i.e., require traversal through a NAT device), or for policy reasons, a domain might require that all traffic between separate NV Domains be funneled through a particular device (e.g., a firewall). In such cases, NVA-2 will advertise to NVA-1 that VM1 on Virtual Network 1217 is available, and direct that traffic between the two nodes go through IP-G. IP-G would then decapsulate received traffic from one NV Domain, translate it appropriately for the other domain and re-encapsulate the packet for delivery.

```
                   xxxxxx                          xxxxxx        +-----+
+-----+     xxxxxxxx    xxxxxx               xxxxxxx     xxxxx   | VM2 |
| VM1 |    xx                xx            xxx               xx  |-----|
|-----|   xx      +           x          xx                  x  |NVE-B|
|NVE-A|   x                   x  +----+  x                    x +-----+
+--+--+   x     NV Domain A   x  |IP-G|--x                     x    |
   +-------x                 xx--+    | x                      xx   |
           x                x    +----+ x      NV Domain B       x   |
        +---x              xx            xx                      x---+
        |    xxxx        xx           +->xx                     xx
        |       xxxxxxxxxx            |   xx                   xx
    +---+-+                           |     xx                xx
    |NVA-1|                       +--+--+    xx           xxx
    +-----+                       |NVA-2|     xxxx     xxxx
                                  +-----+        xxxxxxx
```

Figure 3: VM1 and VM2 are in different NV Domains.

NVAs at one site share information and interact with NVAs at other sites, but only in a controlled manner. It is expected that policy and access control will be applied at the boundaries between different sites (and NVAs) so as to minimize dependencies on external NVAs that could negatively impact the operation within a site. It is an architectural principle that operations involving NVAs at one site not be immediately impacted by failures or errors at another site. (Of course, communication between NVEs in different NV domains may be impacted by such failures or errors.) It is a strong requirement that an NVA continue to operate properly for local NVEs even if external communication is interrupted (e.g., should communication between a local and remote NVA fail).

At a high level, a federation of interconnected NVAs has some analogies to BGP and Autonomous Systems. Like an Autonomous System, NVAs at one site are managed by a single administrative entity and do not interact with external NVAs except as allowed by policy. Likewise, the interface between NVAs at different sites is well defined, so that the internal details of operations at one site are largely hidden to other sites. Finally, an NVA only peers with other

NVAs that it has a trusted relationship with, i.e., where a VN is intended to span multiple NVAs.

Reasons for using a federated model include:

- o Provide isolation among NVAs operating at different sites at different geographic locations.

- o Control the quantity and rate of information updates that flow (and must be processed) between different NVAs in different data centers.

- o Control the set of external NVAs (and external sites) a site peers with. A site will only peer with other sites that are cooperating in providing an overlay service.

- o Allow policy to be applied between sites. A site will want to carefully control what information it exports (and to whom) as well as what information it is willing to import (and from whom).

- o Allow different protocols and architectures to be used to for intra- vs. inter-NVA communication. For example, within a single data center, a replicated transaction server using database techniques might be an attractive implementation option for an NVA, and protocols optimized for intra-NVA communication would likely be different from protocols involving inter-NVA communication between different sites.

- o Allow for optimized protocols, rather than using a one-size-fits all approach. Within a data center, networks tend to have lower-latency, higher-speed and higher redundancy when compared with WAN links interconnecting data centers. The design constraints and tradeoffs for a protocol operating within a data center network are different from those operating over WAN links. While a single protocol could be used for both cases, there could be advantages to using different and more specialized protocols for the intra- and inter-NVA case.

9.1. Inter-NVA Peering

To support peering between different NVAs, an inter-NVA protocol is needed. The inter-NVA protocol defines what information is exchanged between NVAs. It is assumed that the protocol will be used to share addressing information between data centers and must scale well over WAN links.

10. Control Protocol Work Areas

The NVO3 architecture consists of two major distinct entities: NVEs and NVAs. In order to provide isolation and independence between these two entities, the NVO3 architecture calls for well defined protocols for interfacing between them. For an individual NVA, the architecture calls for a logically centralized entity that could be implemented in a distributed or replicated fashion. While the IETF may choose to define one or more specific architectural approaches to building individual NVAs, there is little need for it to pick exactly one approach to the exclusion of others. An NVA for a single domain will likely be deployed as a single vendor product and thus there is little benefit in standardizing the internal structure of an NVA.

Individual NVAs peer with each other in a federated manner. The NVO3 architecture calls for a well-defined interface between NVAs.

Finally, a hypervisor-to-NVE protocol is needed to cover the split-NVE scenario described in Section 4.2.

11. NVO3 Data Plane Encapsulation

When tunneling tenant traffic, NVEs add encapsulation header to the original tenant packet. The exact encapsulation to use for NVO3 does not seem to be critical. The main requirement is that the encapsulation support a Context ID of sufficient size [I-D.ietf-nvo3-dataplane-requirements]. A number of encapsulations already exist that provide a VN Context of sufficient size for NVO3. For example, VXLAN [RFC7348] has a 24-bit VXLAN Network Identifier (VNI). NVGRE [I-D.sridharan-virtualization-nvgre] has a 24-bit Tenant Network ID (TNI). MPLS-over-GRE provides a 20-bit label field. While there is widespread recognition that a 12-bit VN Context would be too small (only 4096 distinct values), it is generally agreed that 20 bits (1 million distinct values) and 24 bits (16.8 million distinct values) are sufficient for a wide variety of deployment scenarios.

12. Operations and Management

The simplicity of operating and debugging overlay networks will be critical for successful deployment. Some architectural choices can facilitate or hinder OAM. Related OAM drafts include [I-D.ashwood-nvo3-operational-requirement].

13. Summary

This document presents the overall architecture for overlays in NVO3. The architecture calls for three main areas of protocol work:

1. A hypervisor-to-NVE protocol to support Split NVEs as discussed in Section 4.2.

2. An NVE to NVA protocol for disseminating VN information (e.g., inner to outer address mappings).

3. An NVA-to-NVA protocol for exchange of information about specific virtual networks between federated NVAs.

It should be noted that existing protocols or extensions of existing protocols are applicable.

14. Acknowledgments

Helpful comments and improvements to this document have come from Lizhong Jin, Anton Ivanov, Dennis (Xiaohong) Qin, Erik Smith, Ziye Yang and Lucy Yong.

15. IANA Considerations

This memo includes no request to IANA.

16. Security Considerations

The NVO3 architecure will need to defend against a number of potential security threats involving both the data plane and control plane.

For the data plane, tunneled application traffic may need protection against being misdelivered, modified, or having its content exposed to an inappropriate third party. In all cases, encryption between authenticated tunnel endpoints can be used to mitigate risks.

For the control plane, between NVAs, the NVA and NVE as well as between different components of the split-NVE approach, a combination of authentication and encryption can be used. All entities will need to properly authenticate with each other and enable encryption for their interactions as appropriate to protect sensitive information.

Leakage of sensitive information about users or other entities associated with VMs whose traffic is virtualized can also be covered by using encryption for the control plane protocols.

17. Informative References

[I-D.ashwood-nvo3-operational-requirement]
Ashwood-Smith, P., Iyengar, R., Tsou, T., Sajassi, A., Boucadair, M., Jacquenet, C., and M. Daikoku, "NVO3 Operational Requirements", draft-ashwood-nvo3-operational-requirement-03 (work in progress), July 2013.

[I-D.ietf-nvo3-dataplane-requirements]
Bitar, N., Lasserre, M., Balus, F., Morin, T., Jin, L., and B. Khasnabish, "NVO3 Data Plane Requirements", draft-ietf-nvo3-dataplane-requirements-03 (work in progress), April 2014.

[I-D.ietf-nvo3-nve-nva-cp-req]
Kreeger, L., Dutt, D., Narten, T., and D. Black, "Network Virtualization NVE to NVA Control Protocol Requirements", draft-ietf-nvo3-nve-nva-cp-req-04 (work in progress), July 2015.

[I-D.sridharan-virtualization-nvgre]
Garg, P. and Y. Wang, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-08 (work in progress), April 2015.

[IEEE-802.1Q]
IEEE 802.1Q-2011, , "IEEE standard for local and metropolitan area networks: Media access control (MAC) bridges and virtual bridged local area networks,", August 2011.

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <http://www.rfc-editor.org/info/rfc826>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <http://www.rfc-editor.org/info/rfc4364>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

[RFC7364] Narten, T., Ed., Gray, E., Ed., Black, D., Fang, L., Kreeger, L., and M. Napierala, "Problem Statement: Overlays for Network Virtualization", RFC 7364, DOI 10.17487/RFC7364, October 2014, <http://www.rfc-editor.org/info/rfc7364>.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, DOI 10.17487/RFC7365, October 2014, <http://www.rfc-editor.org/info/rfc7365>.

Appendix A. Change Log

A.1. Changes From draft-ietf-nvo3-arch-03 to -04

1. First cut at proper Security Considerations Section.

2. Fixed some obvious typos.

A.2. Changes From draft-ietf-nvo3-arch-02 to -03

1. Removed "[Note:" comments from section 7.3 and 8.

2. Removed discussion stimulating "[Note" comment from section 8.1 and changed the text to note that the NVO3 architecture will focus on a model where all NVEs interact with the NVA.

3. Added a subsection on NVO3 Gateway taxonomy.

A.3. Changes From draft-ietf-nvo3-arch-01 to -02

1. Minor editorial improvements after a close re-reading; references to problem statement and framework updated to point to recently-published RFCs.

2. Added text making it more clear that other virtualization approaches, including Linux Containers are intended to be fully supported in NVO3.

A.4. Changes From draft-ietf-nvo3-arch-00 to -01

1. Miscellaneous text/section additions, including:

   * New section on VLAN tag Handling (Section 3.1.1).

   * New section on tenant VLAN handling in Split-NVE case (Section 4.2.1).

   * New section on TTL handling (Section 3.1.2).

   * New section on multi-homing of NVEs (Section 4.4).

   * 2 paragraphs new text describing L2/L3 Combined service (Section 3.1).

   * New section on VAPs (and error handling) (Section 4.5).

   * New section on ARP and ND handling (Section 5.5)

   * New section on NVE-to-NVE interactions (Section 6)

2. Editorial cleanups from careful review by Erik Smith, Ziye Yang.

3. Expanded text on Distributed Inter-VN Gateways.

A.5. Changes From draft-narten-nvo3 to draft-ietf-nvo3

1. No changes between draft-narten-nvo3-arch-01 and draft-ietf-nvoe-arch-00.

A.6. Changes From -00 to -01 (of draft-narten-nvo3-arch)

1. Editorial and clarity improvements.

2. Replaced "push vs. pull" section with section more focused on triggers where an event implies or triggers some action.

3. Clarified text on co-located NVE to show how offloading NVE functionality onto adapters is desirable.

4. Added new section on distributed gateways.

5. Expanded Section on NVA external interface, adding requirement for NVE to support multiple IP NVA addresses.

Authors' Addresses

David Black
EMC

Email: david.black@emc.com

Jon Hudson
Brocade
120 Holger Way
San Jose, CA 95134
USA

Email: jon.hudson@gmail.com

Lawrence Kreeger
Cisco

Email: kreeger@cisco.com

Marc Lasserre
Alcatel-Lucent

Email: marc.lasserre@alcatel-lucent.com

Thomas Narten
IBM

Email: narten@us.ibm.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 9, 2015

J. Gross
T. Sridhar
VMware
P. Garg
Microsoft
C. Wright
Red Hat
I. Ganga
Intel
P. Agarwal
Innovium
K. Duda
Arista
D. Dutt
Cumulus
J. Hudson
Brocade

May 8, 2015

Geneve: Generic Network Virtualization Encapsulation
draft-ietf-nvo3-geneve-00

Abstract

Network virtualization involves the cooperation of devices with a wide variety of capabilities such as software and hardware tunnel endpoints, transit fabrics, and centralized control clusters. As a result of their role in tying together different elements in the system, the requirements on tunnels are influenced by all of these components. Flexibility is therefore the most important aspect of a tunnel protocol if it is to keep pace with the evolution of the system. This draft describes Geneve, a protocol designed to recognize and accommodate these changing capabilities and needs.

Status of This Memo

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 9, 2015.

Copyright Notice

Table of Contents

1. Introduction

Networking has long featured a variety of tunneling, tagging, and other encapsulation mechanisms. However, the advent of network virtualization has caused a surge of renewed interest and a corresponding increase in the introduction of new protocols. The large number of protocols in this space, ranging all the way from VLANs [IEEE.802.1Q-2011] and MPLS [RFC3031] through the more recent VXLAN [RFC7348], NVGRE [I-D.sridharan-virtualization-nvgre], and STT [I-D.davie-stt], often leads to questions about the need for new encapsulation formats and what it is about network virtualization in particular that leads to their proliferation.

While many encapsulation protocols seek to simply partition the underlay network or bridge between two domains, network virtualization views the transit network as providing connectivity between multiple components of an integrated system. In many ways this system is similar to a chassis switch with the IP underlay network playing the role of the backplane and tunnel endpoints on the edge as line cards. When viewed in this light, the requirements placed on the tunnel protocol are significantly different in terms of the quantity of metadata necessary and the role of transit nodes.

Current work such as [VL2] and the NVO3 working group [I-D.ietf-nvo3-dataplane-requirements] have described some of the properties that the data plane must have to support network virtualization. However, one additional defining requirement is the need to carry system state along with the packet data. The use of some metadata is certainly not a foreign concept - nearly all protocols used for virtualization have at least 24 bits of identifier space as a way to partition between tenants. This is often described as overcoming the limits of 12-bit VLANs, and when seen in that context, or any context where it is a true tenant identifier, 16 million possible entries is a large number. However, the reality is that the metadata is not exclusively used to identify tenants and encoding other information quickly starts to crowd the space. In fact, when compared to the tags used to exchange metadata between line cards on a chassis switch, 24-bit identifiers start to look quite small. There are nearly endless uses for this metadata, ranging from storing input ports for simple security policies to service based context for interposing advanced middleboxes.

Existing tunnel protocols have each attempted to solve different aspects of these new requirements, only to be quickly rendered out of date by changing control plane implementations and advancements. Furthermore, software and hardware components and controllers all have different advantages and rates of evolution - a fact that should be viewed as a benefit, not a liability or limitation. This draft describes Geneve, a protocol which seeks to avoid these problems by providing a framework for tunneling for network virtualization rather than being prescriptive about the entire system.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

1.2. Terminology

The NVO3 framework [RFC7365] defines many of the concepts commonly used in network virtualization. In addition, the following terms are specifically meaningful in this document:

Checksum offload. An optimization implemented by many NICs which enables computation and verification of upper layer protocol checksums in hardware on transmit and receive, respectively. This typically includes IP and TCP/UDP checksums which would otherwise be computed by the protocol stack in software.

Clos network. A technique for composing network fabrics larger than a single switch while maintaining non-blocking bandwidth across connection points. ECMP is used to divide traffic across the multiple links and switches that constitute the fabric. Sometimes termed "leaf and spine" or "fat tree" topologies.

ECMP. Equal Cost Multipath. A routing mechanism for selecting from among multiple best next hop paths by hashing packet headers in order to better utilize network bandwidth while avoiding reordering a single stream.

Geneve. Generic Network Virtualization Encapsulation. The tunnel protocol described in this draft.

LRO. Large Receive Offload. The receive-side equivalent function of LSO, in which multiple protocol segments (primarily TCP) are coalesced into larger data units.

NIC. Network Interface Card. A NIC could be part of a tunnel endpoint or transit device and can either process Geneve packets or aid in the processing of Geneve packets.

OAM. Operations, Administration, and Management. A suite of tools used to monitor and troubleshoot network problems.

Transit device. A forwarding element along the path of the tunnel making up part of the Underlay Network. A transit device MAY be capable of understanding the Geneve frame format but does not originate or terminate Geneve packets.

LSO. Large Segmentation Offload. A function provided by many commercial NICs that allows data units larger than the MTU to be passed to the NIC to improve performance, the NIC being responsible for creating smaller segments with correct protocol headers. When referring specifically to TCP/IP, this feature is often known as TSO (TCP Segmentation Offload).

Tunnel endpoint. A component encapsulating packets, such as Ethernet frames or IP datagrams, in Geneve headers and vice versa. As the ultimate consumer of any tunnel metadata, endpoints have the highest level of requirements for parsing and interpreting tunnel headers. Tunnel endpoints may consist of either software or hardware implementations or a combination of the two. Endpoints are frequently a component of an NVE but may also be found in middleboxes or other elements making up an NVO3 Network.

VM. Virtual Machine.

2. Design Requirements

Geneve is designed to support network virtualization use cases, where tunnels are typically established to act as a backplane between the virtual switches residing in hypervisors, physical switches, or middleboxes or other appliances. An arbitrary IP network can be used as an underlay although Clos networks composed using ECMP links are a common choice to provide consistent bisectional bandwidth across all connection points. Figure 1 shows an example of a hypervisor, top of rack switch for connectivity to physical servers, and a WAN uplink connected using Geneve tunnels over a simplified Clos network. These tunnels are used to encapsulate and forward frames from the attached components such as VMs or physical links.

```
     +---------------------+           +-------+  +------+
     | +--+  +-------+---+ |           |Transit|--|Top of|==Physical
     | |VM|--|       |   | | +------+ /|Router |  | Rack |==Servers
     | +--+  |Virtual|NIC|---|Top of|/ +-------+\/+------+
     | +--+  |Switch |   | | | Rack |\ +-------+/\+------+
     | |VM|--|       |   | | +------+ \|Transit|  |Uplink|   WAN
     | +--+  +-------+---+ |           |Router |--|      |=========>
     +---------------------+           +-------+  +------+
            Hypervisor

                 ()===================================()
                         Switch-Switch Geneve Tunnels
```

Figure 1: Sample Geneve Deployment

To support the needs of network virtualization, the tunnel protocol should be able to take advantage of the differing (and evolving) capabilities of each type of device in both the underlay and overlay networks. This results in the following requirements being placed on the data plane tunneling protocol:

- o The data plane is generic and extensible enough to support current and future control planes.

- o Tunnel components are efficiently implementable in both hardware and software without restricting capabilities to the lowest common denominator.

- o High performance over existing IP fabrics.

These requirements are described further in the following subsections.

2.1. Control Plane Independence

Although some protocols for network virtualization have included a control plane as part of the tunnel format specification (most notably, the original VXLAN spec prescribed a multicast learning-based control plane), these specifications have largely been treated as describing only the data format. The VXLAN frame format has actually seen a wide variety of control planes built on top of it.

There is a clear advantage in settling on a data format: most of the protocols are only superficially different and there is little advantage in duplicating effort. However, the same cannot be said of control planes, which are diverse in very fundamental ways. The case for standardization is also less clear given the wide variety in requirements, goals, and deployment scenarios.

As a result of this reality, Geneve aims to be a pure tunnel format specification that is capable of fulfilling the needs of many control planes by explicitly not selecting any one of them. This simultaneously promotes a shared data format and increases the chances that it will not be obsoleted by future control plane enhancements.

2.2. Data Plane Extensibility

Achieving the level of flexibility needed to support current and future control planes effectively requires an options infrastructure to allow new metadata types to be defined, deployed, and either finalized or retired. Options also allow for differentiation of products by encouraging independent development in each vendor’s core specialty, leading to an overall faster pace of advancement. By far the most common mechanism for implementing options is Type-Length-Value (TLV) format.

It should be noted that while options can be used to support non-wirespeed control frames, they are equally important on data frames as well to segregate and direct forwarding (for instance, the examples given before of input port based security policies and service interposition both require tags to be placed on data packets). Therefore, while it would be desirable to limit the extensibility to only control frames for the purposes of simplifying the datapath, that would not satisfy the design requirements.

2.2.1. Efficient Implementation

There is often a conflict between software flexibility and hardware performance that is difficult to resolve. For a given set of functionality, it is obviously desirable to maximize performance. However, that does not mean new features that cannot be run at that speed today should be disallowed. Therefore, for a protocol to be efficiently implementable means that a set of common capabilities can be reasonably handled across platforms along with a graceful mechanism to handle more advanced features in the appropriate situations.

The use of a variable length header and options in a protocol often raises questions about whether it is truly efficiently implementable in hardware. To answer this question in the context of Geneve, it is important to first divide "hardware" into two categories: tunnel endpoints and transit devices.

Endpoints must be able to parse the variable header, including any options, and take action. Since these devices are actively participating in the protocol, they are the most affected by Geneve.

However, as endpoints are the ultimate consumers of the data, transmitters can tailor their output to the capabilities of the recipient. As new functionality becomes sufficiently well defined to add to endpoints, supporting options can be designed using ordering restrictions and other techniques to ease parsing.

Transit devices MAY be able to interpret the options and participate in Geneve packet processing. However, as non-terminating devices, they do not originate or terminate the Geneve packet. The participation of transit devices in Geneve packet processing is OPTIONAL.

Further, either tunnel endpoints or transit devices MAY use offload capabilities of NICs such as checksum offload to improve the performance of Geneve packet processing. The presence of a Geneve variable length header SHOULD NOT prevent the tunnel endpoints and transit devices from using such offload capabilities.

2.3. Use of Standard IP Fabrics

IP has clearly cemented its place as the dominant transport mechanism and many techniques have evolved over time to make it robust, efficient, and inexpensive. As a result, it is natural to use IP fabrics as a transit network for Geneve. Fortunately, the use of IP encapsulation and addressing is enough to achieve the primary goal of delivering packets to the correct point in the network through standard switching and routing.

In addition, nearly all underlay fabrics are designed to exploit parallelism in traffic to spread load across multiple links without introducing reordering in individual flows. These equal cost multipathing (ECMP) techniques typically involve parsing and hashing the addresses and port numbers from the packet to select an outgoing link. However, the use of tunnels often results in poor ECMP performance without additional knowledge of the protocol as the encapsulated traffic is hidden from the fabric by design and only endpoint addresses are available for hashing.

Since it is desirable for Geneve to perform well on these existing fabrics, it is necessary for entropy from encapsulated packets to be exposed in the tunnel header. The most common technique for this is to use the UDP source port, which is discussed further in Section 3.3.

3. Geneve Encapsulation Details

The Geneve frame format consists of a compact tunnel header encapsulated in UDP over either IPv4 or IPv6. A small fixed tunnel header provides control information plus a base level of functionality and interoperability with a focus on simplicity. This header is then followed by a set of variable options to allow for future innovation. Finally, the payload consists of a protocol data unit of the indicated type, such as an Ethernet frame. The following subsections provide examples of Geneve frames transported (for example) over Ethernet along with an Ethernet payload.

3.1. Geneve Frame Format Over IPv4

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Outer Ethernet Header:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Outer Destination MAC Address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Outer Destination MAC Address |   Outer Source MAC Address    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Outer Source MAC Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Optional Ethertype=C-Tag 802.1Q|  Outer VLAN Tag Information   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Ethertype=0x0800        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Outer IPv4 Header:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |Protocol=17 UDP|         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Outer Source IPv4 Address               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Outer Destination IPv4 Address              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Outer UDP Header:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Source Port = xxxx      |       Dest Port = 6081        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           UDP Length          |        UDP Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Geneve Header:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|  Opt Len  |O|C|    Rsvd.  |          Protocol Type        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Virtual Network Identifier (VNI)       |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Variable Length Options                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Inner Ethernet Header (example payload):

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Inner Destination MAC Address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Inner Destination MAC Address |   Inner Source MAC Address    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Inner Source MAC Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Optional Ethertype=C-Tag 802.1Q|  Inner VLAN Tag Information  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Payload:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Ethertype of Original Payload |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                 Original Ethernet Payload     |
|                                                               |
| (Note that the original Ethernet Frame's FCS is not included) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Frame Check Sequence:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   New FCS (Frame Check Sequence) for Outer Ethernet Frame     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2. Geneve Frame Format Over IPv6

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Outer Ethernet Header:
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                 Outer Destination MAC Address                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Outer Destination MAC Address |   Outer Source MAC Address    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Outer Source MAC Address                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Optional Ethertype=C-Tag 802.1Q|  Outer VLAN Tag Information   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       Ethertype=0x86DD        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Outer IPv6 Header:
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version| Traffic Class |           Flow Label                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Payload Length        | NxtHdr=17 UDP |   Hop Limit   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   |                                                               |
   +                     Outer Source IPv6 Address                 +
   |                                                               |
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   |                                                               |
   +                   Outer Destination IPv6 Address              +
   |                                                               |
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Outer UDP Header:
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       Source Port = xxxx      |       Dest Port = 6081        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           UDP Length          |        UDP Checksum           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Geneve Header:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|  Opt Len  |O|C|    Rsvd.  |          Protocol Type        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Virtual Network Identifier (VNI)       |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Variable Length Options                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Inner Ethernet Header (example payload):

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Inner Destination MAC Address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Inner Destination MAC Address |   Inner Source MAC Address    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Inner Source MAC Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Optional Ethertype=C-Tag 802.1Q|  Inner VLAN Tag Information   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Payload:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Ethertype of Original Payload |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                 Original Ethernet Payload     |
|                                                               |
| (Note that the original Ethernet Frame’s FCS is not included) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Frame Check Sequence:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   New FCS (Frame Check Sequence) for Outer Ethernet Frame     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.3. UDP Header

The use of an encapsulating UDP [RFC0768] header follows the connectionless semantics of Ethernet and IP in addition to providing entropy to routers performing ECMP. The header fields are therefore interpreted as follows:

Source port: A source port selected by the ingress tunnel endpoint. This source port SHOULD be the same for all packets belonging to a single encapsulated flow to prevent reordering due to the use of different paths. To encourage an even distribution of flows across multiple links, the source port SHOULD be calculated using a hash of the encapsulated packet headers using, for example, a traditional 5-tuple. Since the port represents a flow identifier

rather than a true UDP connection, the entire 16-bit range MAY be used to maximize entropy.

Dest port: IANA has assigned port 6081 as the fixed well-known destination port for Geneve. This port MUST be used in both directions of a flow. Although the well-known value should be used by default, it is RECOMMENDED that implementations make this configurable.

UDP length: The length of the UDP packet including the UDP header.

UDP checksum: The checksum MAY be set to zero on transmit for packets encapsulated in both IPv4 and IPv6 [RFC6935]. When a packet is received with a UDP checksum of zero it MUST be accepted and decapsulated. If the ingress tunnel endpoint optionally encapsulates a packet with a non-zero checksum, it MUST be a correctly computed UDP checksum. Upon receiving such a packet, the egress endpoint MUST validate the checksum. If the checksum is not correct, the packet MUST be dropped, otherwise the packet MUST be accepted for decapsulation. It is RECOMMENDED that the UDP checksum be computed to protect the Geneve header and options in situations where the network reliability is not high and the packet is not protected by another checksum or CRC.

3.4. Tunnel Header Fields

Ver (2 bits): The current version number is 0. Packets received by an endpoint with an unknown version MUST be dropped. Non-terminating devices processing Geneve packets with an unknown version number MUST treat them as UDP packets with an unknown payload.

Opt Len (6 bits): The length of the options fields, expressed in four byte multiples, not including the eight byte fixed tunnel header. This results in a minimum total Geneve header size of 8 bytes and a maximum of 260 bytes. The start of the payload headers can be found using this offset from the end of the base Geneve header.

O (1 bit): OAM frame. This packet contains a control message instead of a data payload. Endpoints MUST NOT forward the payload and transit devices MUST NOT attempt to interpret or process it. Since these are infrequent control messages, it is RECOMMENDED that endpoints direct these packets to a high priority control queue (for example, to direct the packet to a general purpose CPU from a forwarding ASIC or to separate out control traffic on a NIC). Transit devices MUST NOT alter forwarding behavior on the basis of this bit, such as ECMP link selection.

C (1 bit): Critical options present. One or more options has the critical bit set (see Section 3.5). If this bit is set then tunnel endpoints MUST parse the options list to interpret any critical options. On devices where option parsing is not supported the frame MUST be dropped on the basis of the 'C' bit in the base header. If the bit is not set tunnel endpoints MAY strip all options using 'Opt Len' and forward the decapsulated frame. Transit devices MUST NOT drop or modify packets on the basis of this bit.

Rsvd. (6 bits): Reserved field which MUST be zero on transmission and ignored on receipt.

Protocol Type (16 bits): The type of the protocol data unit appearing after the Geneve header. This follows the EtherType [ETYPES] convention with Ethernet itself being represented by the value 0x6558.

Virtual Network Identifier (VNI) (24 bits): An identifier for a unique element of a virtual network. In many situations this may represent an L2 segment, however, the control plane defines the forwarding semantics of decapsulated packets. The VNI MAY be used as part of ECMP forwarding decisions or MAY be used as a mechanism to distinguish between overlapping address spaces contained in the encapsulated packet when load balancing across CPUs.

Reserved (8 bits): Reserved field which MUST be zero on transmission and ignored on receipt.

Transit devices MUST maintain consistent forwarding behavior irrespective of the value of 'Opt Len', including ECMP link selection. These devices SHOULD be able to forward packets containing options without resorting to a slow path.

3.5. Tunnel Options

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Option Class         |      Type     |R|R|R| Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Variable Option Data                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Geneve Option

The base Geneve header is followed by zero or more options in Type-Length-Value format. Each option consists of a four byte option

header and a variable amount of option data interpreted according to the type.

Option Class (16 bits): Namespace for the 'Type' field. IANA will be requested to create a "Geneve Option Class" registry to allocate identifiers for organizations, technologies, and vendors that have an interest in creating types for options. Each organization may allocate types independently to allow experimentation and rapid innovation. It is expected that over time certain options will become well known and a given implementation may use option types from a variety of sources. In addition, IANA will be requested to reserve specific ranges for standardized and experimental options.

Type (8 bits): Type indicating the format of the data contained in this option. Options are primarily designed to encourage future extensibility and innovation and so standardized forms of these options will be defined in a separate document.

The high order bit of the option type indicates that this is a critical option. If the receiving endpoint does not recognize this option and this bit is set then the frame MUST be dropped. If the critical bit is set in any option then the 'C' bit in the Geneve base header MUST also be set. Transit devices MUST NOT drop packets on the basis of this bit. The following figure shows the location of the 'C' bit in the 'Type' field:

```
0 1 2 3 4 5 6 7 8
+-+-+-+-+-+-+-+-+
|C|    Type     |
+-+-+-+-+-+-+-+-+
```

The requirement to drop a packet with an unknown critical option applies to the entire tunnel endpoint system and not a particular component of the implementation. For example, in a system comprised of a forwarding ASIC and a general purpose CPU, this does not mean that the packet must be dropped in the ASIC. An implementation may send the packet to the CPU using a rate-limited control channel for slow-path exception handling.

R (3 bits): Option control flags reserved for future use. MUST be zero on transmission and ignored on receipt.

Length (5 bits): Length of the option, expressed in four byte multiples excluding the option header. The total length of each option may be between 4 and 128 bytes. Packets in which the total length of all options is not equal to the 'Opt Len' in the base

header are invalid and MUST be silently dropped if received by an endpoint.

Variable Option Data: Option data interpreted according to 'Type'.

3.5.1. Options Processing

Geneve options are intended to be originated and processed by tunnel endpoints. Options MAY be processed by transit devices along the tunnel path as well. This document only details the handling of options by tunnel endpoints. A future version of this document will provide details of options processing by transit devices. Transit devices not processing Geneve options SHOULD process Geneve frame as any other UDP frame and maintain consistent forwarding behavior.

In tunnel endpoints, the generation and interpretation of options is determined by the control plane, which is out of the scope of this document. However, to ensure interoperability between heterogeneous devices two requirements are imposed on endpoint devices:

- o Receiving endpoints MUST drop packets containing unknown options with the 'C' bit set in the option type.

- o Sending endpoints MUST NOT assume that options will be processed sequentially by the receiver in the order they were transmitted.

4. Implementation and Deployment Considerations

4.1. Encapsulation of Geneve in IP

As an IP-based tunnel protocol, Geneve shares many properties and techniques with existing protocols. The application of some of these are described in further detail, although in general most concepts applicable to the IP layer or to IP tunnels generally also function in the context of Geneve.

4.1.1. IP Fragmentation

To prevent fragmentation and maximize performance, the best practice when using Geneve is to ensure that the MTU of the physical network is greater than or equal to the MTU of the encapsulated network plus tunnel headers. Manual or upper layer (such as TCP MSS clamping) configuration can be used to ensure that fragmentation never takes place, however, in some situations this may not be feasible.

It is strongly RECOMMENDED that Path MTU Discovery ([RFC1191], [RFC1981]) be used by setting the DF bit in the IP header when Geneve packets are transmitted over IPv4 (this is the default with IPv6).

The use of Path MTU Discovery on the transit network provides the encapsulating endpoint with soft-state about the link that it may use to prevent or minimize fragmentation depending on its role in the virtualized network.

Note that some implementations may not be capable of supporting fragmentation or other less common features of the IP header, such as options and extension headers.

4.1.2. DSCP and ECN

When encapsulating IP (including over Ethernet) frames in Geneve, there are several options for propagating DSCP and ECN bits from the inner header to the tunnel on transmission and the reverse on reception.

[RFC2983] lists considerations for mapping DSCP between inner and outer IP headers. Network virtualization is typically more closely aligned with the Pipe model described, where the DSCP value on the tunnel header is set based on a policy (which may be a fixed value, one based on the inner traffic class, or some other mechanism for grouping traffic). Aspects of the Uniform model (which treats the inner and outer DSCP value as a single field by copying on ingress and egress) may also apply, such as the ability to remark the inner header on tunnel egress based on transit marking. However, the Uniform model is not conceptually consistent with network virtualization, which seeks to provide strong isolation between encapsulated traffic and the physical network.

[RFC6040] describes the mechanism for exposing ECN capabilities on IP tunnels and propagating congestion markers to the inner packets. This behavior SHOULD be followed for IP packets encapsulated in Geneve.

4.1.3. Broadcast and Multicast

Geneve tunnels may either be point-to-point unicast between two endpoints or may utilize broadcast or multicast addressing. It is not required that inner and outer addressing match in this respect. For example, in physical networks that do not support multicast, encapsulated multicast traffic may be replicated into multiple unicast tunnels or forwarded by policy to a unicast location (possibly to be replicated there).

With physical networks that do support multicast it may be desirable to use this capability to take advantage of hardware replication for encapsulated packets. In this case, multicast addresses may be allocated in the physical network corresponding to tenants,

encapsulated multicast groups, or some other factor. The allocation of these groups is a component of the control plane and therefore outside of the scope of this document. When physical multicast is in use, the 'C' bit in the Geneve header may be used with groups of devices with heterogeneous capabilities as each device can interpret only the options that are significant to it if they are not critical.

4.2. NIC Offloads

Modern NICs currently provide a variety of offloads to enable the efficient processing of packets. The implementation of many of these offloads requires only that the encapsulated packet be easily parsed (for example, checksum offload). However, optimizations such as LSO and LRO involve some processing of the options themselves since they must be replicated/merged across multiple packets. In these situations, it is desirable to not require changes to the offload logic to handle the introduction of new options. To enable this, some constraints are placed on the definitions of options to allow for simple processing rules:

- When performing LSO, a NIC MUST replicate the entire Geneve header and all options, including those unknown to the device, onto each resulting segment. However, a given option definition may override this rule and specify different behavior in supporting devices. Conversely, when performing LRO, a NIC MAY assume that a binary comparison of the options (including unknown options) is sufficient to ensure equality and MAY merge packets with equal Geneve headers.

- Option ordering is not significant and packets with the same options in a different order MAY be processed alike.

- NICs performing offloads MUST NOT drop packets with unknown options, including those marked as critical.

There is no requirement that a given implementation of Geneve employ the offloads listed as examples above. However, as these offloads are currently widely deployed in commercially available NICs, the rules described here are intended to enable efficient handling of current and future options across a variety of devices.

4.3. Inner VLAN Handling

Geneve is capable of encapsulating a wide range of protocols and therefore a given implementation is likely to support only a small subset of the possibilities. However, as Ethernet is expected to be widely deployed, it is useful to describe the behavior of VLANs inside encapsulated Ethernet frames.

As with any protocol, support for inner VLAN headers is OPTIONAL. In many cases, the use of encapsulated VLANs may be disallowed due to security or implementation considerations. However, in other cases trunking of VLAN frames across a Geneve tunnel can prove useful. As a result, the processing of inner VLAN tags upon ingress or egress from a tunnel endpoint is based upon the configuration of the endpoint and/or control plane and not explicitly defined as part of the data format.

5. Interoperability Issues

Viewed exclusively from the data plane, Geneve does not introduce any interoperability issues as it appears to most devices as UDP frames. However, as there are already a number of tunnel protocols deployed in network virtualization environments, there is a practical question of transition and coexistence.

Since Geneve is a superset of the functionality of the three most common protocols used for network virtualization (VXLAN, NVGRE, and STT) it should be straightforward to port an existing control plane to run on top of it with minimal effort. With both the old and new frame formats supporting the same set of capabilities, there is no need for a hard transition - endpoints directly communicating with each other use any common protocol, which may be different even within a single overall system. As transit devices are primarily forwarding frames on the basis of the IP header, all protocols appear similar and these devices do not introduce additional interoperability concerns.

To assist with this transition, it is strongly suggested that implementations support simultaneous operation of both Geneve and existing tunnel protocols as it is expected to be common for a single node to communicate with a mixture of other nodes. Eventually, older protocols may be phased out as they are no longer in use.

6. Security Considerations

As UDP/IP packets, Geneve does not have any inherent security mechanisms. As a result, an attacker with access to the underlay network transporting the IP frames has the ability to snoop or inject packets. Legitimate but malicious tunnel endpoints may also spoof identifiers in the tunnel header to gain access to networks owned by other tenants.

Within a particular security domain, such as a data center operated by a single provider, the most common and highest performing security mechanism is isolation of trusted components. Tunnel traffic can be carried over a separate VLAN and filtered at any untrusted

boundaries. In addition, tunnel endpoints should only be operated in environments controlled by the service provider, such as the hypervisor itself rather than within a customer VM.

When crossing an untrusted link, such as the public Internet, IPsec [RFC4301] may be used to provide authentication and/or encryption of the IP packets. If the remote tunnel endpoint is not completely trusted, for example it resides on a customer premises, then it may also be necessary to sanitize any tunnel metadata to prevent tenant-hopping attacks.

7. IANA Considerations

IANA has allocated UDP port 6081 as the well-known destination port for Geneve. Upon publication, the registry should be updated to cite this document. The original request was:

Service Name: geneve
Transport Protocol(s): UDP
Assignee: Jesse Gross <jgross@vmware.com>
Contact: Jesse Gross <jgross@vmware.com>
Description: Generic Network Virtualization Encapsulation (Geneve)
Reference: This document
Port Number: 6081

In addition, IANA is requested to create a "Geneve Option Class" registry to allocate Option Classes. This shall be a registry of 16-bit hexadecimal values along with descriptive strings. The identifiers 0x0-0xFF are to be reserved for standardized options for allocation by IETF Review [RFC5226] and 0xFFFF for Experimental Use. Otherwise, identifiers are to be assigned to any organization with an interest in creating Geneve options on a First Come First Served basis. There are no initial registry assignments.

8. Acknowledgements

The authors wish to thank Martin Casado, Bruce Davie and Dave Thaler for their input, feedback, and helpful suggestions.

9. References

9.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

9.2. Informative References

[ETYPES] The IEEE Registration Authority, "IEEE 802 Numbers", 2013, <http://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xml>.

[I-D.davie-stt] Davie, B. and J. Gross, "A Stateless Transport Tunneling Protocol for Network Virtualization (STT)", draft-davie-stt-06 (work in progress), April 2014.

[I-D.ietf-nvo3-dataplane-requirements] Bitar, N., Lasserre, M., Balus, F., Morin, T., Jin, L., and B. Khasnabish, "NVO3 Data Plane Requirements", draft-ietf-nvo3-dataplane-requirements-03 (work in progress), April 2014.

[I-D.sridharan-virtualization-nvgre] Garg, P. and Y. Wang, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-08 (work in progress), April 2015.

[IEEE.802.1Q-2011] IEEE, "IEEE Standard for Local and metropolitan area networks -- Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks", IEEE Std 802.1Q, 2011.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, April 2013.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, October 2014.

[VL2] Greenberg et al, , "VL2: A Scalable and Flexible Data Center Network", 2009.

Proc. ACM SIGCOMM 2009

Authors' Addresses

Jesse Gross
VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
USA

Email: jgross@vmware.com

T. Sridhar
VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
USA

Email: tsridhar@vmware.com

Pankaj Garg
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052
USA

Email: pankajg@microsoft.com

Chris Wright
Red Hat Inc.
1801 Varsity Drive
Raleigh, NC 27606
USA

Email: chrisw@redhat.com

Ilango Ganga
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
USA

Email: ilango.s.ganga@intel.com

Puneet Agarwal
Innovium, Inc.
6001 America Center Drive
San Jose, CA 95002
USA

Email: puneet@innovium.com

Kenneth Duda
Arista Networks
5453 Great America Parkway
Santa Clara, CA 95054
USA

Email: kduda@arista.com

Dinesh G. Dutt
Cumulus Networks
140C S. Whisman Road
Mountain View, CA 94041
USA

Email: ddutt@cumulusnetworks.com

Jon Hudson
Brocade Communications Systems, Inc.
130 Holger Way
San Jose, CA 95134
USA

Email: jon.hudson@gmail.com

Network Virtualization Overlays (nvo3)
Internet-Draft
Intended status: Standard track
Expires December 14, 2015

T. Herbert
Facebook
L. Yong
Huawei USA
O. Zia
Microsoft

June 24, 2015

Generic UDP Encapsulation
draft-ietf-nvo3-gue-01

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on September 7, 2015.

Copyright Notice

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents

Abstract

This specification describes Generic UDP Encapsulation (GUE), which is a scheme for using UDP to encapsulate packets of arbitrary IP protocols for transport across layer 3 networks. By encapsulating packets in UDP, specialized capabilities in networking hardware for efficient handling of UDP packets can be leveraged. GUE specifies basic encapsulation methods upon which higher level constructs, such tunnels and overlay networks for network virtualization, can be constructed. GUE is extensible by allowing optional data fields as part of the encapsulation, and is generic in that it can encapsulate packets of various IP protocols.

Table of Contents

1. Introduction

This specification describes Generic UDP Encapsulation (GUE) which is a general method for encapsulating packets of arbitrary IP protocols within User Datagram Protocol (UDP) [RFC0768] packets. Encapsulating packets in UDP facilitates efficient transport across networks. Networking devices widely provide protocol specific processing and optimizations for UDP (as well as TCP) packets. Packets for atypical IP protocols (those not usually parsed by networking hardware) can be encapsulated in UDP packets to maximize deliverability and to leverage flow specific mechanisms for routing and packet steering.

GUE provides an extensible header format for including optional data in the encapsulation header. This data potentially covers items such as virtual networking identifier, security data for validating or authenticating the GUE header, congestion control data, etc. GUE also allows private optional data in the encapsulation header. This feature can be used by a site or implementation to define local custom optional data, and allows experimentation of options that may eventually become standard.

2. Packet formats

A GUE packet is comprised of a UDP packet whose payload is a GUE header followed by a payload which is either an encapsulated packet of some IP protocol or a control message (like an OAM message). A GUE packet has the general format:

```
+-------------------------------+
|                               |
|        UDP/IP header          |
|                               |
|-------------------------------|
|                               |
|         GUE Header            |
|                               |
|-------------------------------|
|                               |
|      Encapsulated packet      |
|      or control message       |
|                               |
+-------------------------------+
```

The GUE header is variable length as determined by the presence of optional fields.

2.1. GUE version

The first two bits of the GUE header contain the GUE protocol version number. The rest of the fields after the GUE version number are defined based on the version number. The remainder of this specification describes version 0x0 of GUE.

2.2. GUE header

The header format for version 0x0 of GUE in UDP is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Source port            |      Destination port         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Length              |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0x0|C|   Hlen  |  Proto/ctype  |             Flags             |E|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                       Fields (optional)                       ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Extension flags (optional)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                     Extension fields (optional)               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                     Private data (optional)                   ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The contents of the UDP header are:

- o Source port (inner flow identifier): This should be set to a value that represents the encapsulated flow. The properties of the inner flow identifier are described below.

- o Destination port: The GUE assigned port number, 6080.

- o Length: Canonical length of the UDP packet (length of UDP header and payload).

- o Checksum: Standard UDP checksum (see section 4).

The GUE header consists of:

- o Ver: GUE protocol version (0x0).

- o C: Control flag. When set indicates a control message, not set indicates a data message.

- o Hlen: Length in 32-bit words of the GUE header, including optional fields but not the first four bytes of the header. Computed as (header_len - 4) / 4. All GUE headers are a multiple of four bytes in length. Maximum header length is 128 bytes.

- o Proto/ctype: When the C bit is set this field contains a control message type for the payload. When C bit is not set, the field holds the IP protocol number for the encapsulated packet in the payload. The control message or encapsulated packet begins at the offset provided by Hlen.

- o Flags. Header flags that may be allocated for various purposes and may indicate presence of optional fields. Undefined header flag bits must be set to zero on transmission.

- o 'E' Extension flag. Indicates presence of extension flags option in the optional fields.

- o Fields: Optional fields whose presence is indicated by corresponding flags.

- o Extension flags: An optional field indicated by the E bit. This field provides an additional set of thirty-two bits for flags.

- o Extension fields: Optional fields whose presence is indicated by corresponding extension flags.

- o Private data: Optional private data. If private data is present it immediately follows that last field present in the header. The length of this data is determined by subtracting the starting offset from the header length.

2.3. Proto/ctype field

When the C bit is not set, the proto/ctype field must be set to a valid IP protocol number. The IP protocol number serves as an indication of the type of next protocol header which is contained in the GUE payload at the offset indicated in Hlen. Intermediate devices may parse the GUE payload per the IP protocol number in the proto/ctype field, and header flags cannot affect the interpretation of the proto/ctype field.

IP protocol number 59 ("No next header") may be set to indicate that the GUE payload does not begin with the header of an IP protocol. This would be the case, for instance, if the GUE payload were a fragment when performing GUE level fragmentation. The interpretation of the payload is performed though other means (such as flags and optional fields), and intermediate devices must not parse packets the

packet based on the IP protocol number in this case.

When the C bit is set, the proto/ctype field must be set to a valid control message type. A value of zero indicates that the GUE payload requires further interpretation to deduce the control type. This might be the case when the payload is a fragment of a control message, where only the reassembled packet can be interpreted as a control message.

2.4. Flags and optional fields

Flags and associated optional fields are the primary mechanism of extensibility in GUE. There are sixteen flag bits in the primary GUE header with one being reserved to indicate that an optional extension flags field is present. The extension flags field contains an additional thirty-two flag bits.

A flag may indicate presence of optional fields. The size of an optional field indicated by a flag must be fixed.

Flags may be paired together to allow different lengths for an optional field. For example, if two flag bits are paired, a field may possibly be three different lengths. Regardless of how flag bits may be paired, the lengths and offsets of optional fields corresponding to a set of flags must be well defined.

Optional fields are placed in order of the flags. New flags should be allocated from high to low order bit contiguously without holes. Flags allow random access, for instance to inspect the field corresponding to the Nth flag bit, an implementation only considers the previous N-1 flags to determine the offset. Flags after the Nth flag are not pertinent in calculating the offset of the Nth flag.

Flags (or paired flags) are idempotent such that new flags should not cause reinterpretation of old flags. Also, new flags should not alter interpretation of other elements in the GUE header nor how the message is parsed (for instance, in a data message the proto/ctype field always holds an IP protocol number as an invariant).

2.5. Private data

An implementation may use private data for its own use. The private data immediately follows the last field in the GUE header and is not a fixed length. This data is considered part of the GUE header and must be accounted for in header length (Hlen). The length of the private data must be a multiple of four and is determined by subtracting the offset of private data in the GUE header from the header length. Specifically:

```
Private_length = (Hlen * 4) - Length(flags)
```

Where "Length(flags)" returns the sum of lengths of all the optional fields present in the GUE header. When there is no private data present, length of the private data is zero.

The semantics and interpretation of private data are implementation specific. An encapsulator and decapsulator MUST agree on the meaning of private data before using it. The private data may be structured as necessary, for instance it might contain its own set of flags and optional fields.

If a decapsulator receives a GUE packet with private data, it MUST validate the private data appropriately. If a decapsulator does not expect private data from an encapsulator the packet MUST be dropped. If a decapsulator cannot validate the contents of private data per the provided semantics the packet MUST also be dropped. An implementation may place security data in GUE private data which must be verified for packet acceptance.

3. Message types

3.1. Control messages

Control messages are indicated in the GUE header when the C bit is set. The payload is interpreted as a control message with type specified in the proto/ctype field. The format and contents of the control message are indicated by the type and can be variable length.

Other than interpreting the proto/ctype field as a control message type, the meaning and semantics of the rest of the elements in the GUE header are the same as that of data messages. Forwarding and routing of control messages should be the same as that of a data message with the same outer IP and UDP header and GUE flags-- this ensures that control messages can be created that follow the same path as data messages.

Control messages can be defined for OAM type messages. For instance, an echo request and corresponding echo reply message may be defined to test for liveness.

3.2. Data messages

Data messages are indicated in GUE header with C bit not set. The payload of a data message is interpreted as an encapsulated packet of an IP protocol indicated in the proto/ctype field. The packet immediately follows the GUE header.

Data messages are a primary means of encapsulation and can be used to create tunnels for overlay networks.

4. Operation

The figure below illustrates the use of GUE encapsulation between two servers. Sever 1 is sending packets to server 2. An encapsulator performs encapsulation of packets from server 1. These encapsulated packets traverse the network as UDP packets. At the decapsulator, packets are decapsulated and sent on to server 2. Packet flow in the reverse direction need not be symmetric; GUE encapsulation is not required in the reverse path.

```
+---------------+                       +---------------+
|               |                       |               |
|   Server 1    |                       |    Server 2   |
|               |                       |               |
+---------------+                       +---------------+
       |                                        ^
       V                                        |
+---------------+   +---------------+   +---------------+
|               |   |               |   |               |
| Encapsulator  |-->|    Layer 3    |-->| Decapsulator  |
|               |   |    Network    |   |               |
+---------------+   +---------------+   +---------------+
```

The encapsulator and decapsulator may be co-resident with the corresponding servers, or may be on separate nodes in the network.

4.1. Network tunnel encapsulation

Network tunneling can be achieved by encapsulating layer 2 or layer 3 packets. In this case the encapsulator and decapsulator nodes are the tunnel endpoints. These could be routers that provide network tunnels on behalf of communicating servers.

4.2. Transport layer encapsulation

When encapsulating layer 4 packets, the encapsulator and decapsulator should be co-resident with the servers. In this case, the encapsulation headers are inserted between the IP header and the transport packet. The addresses in the IP header refer to both the endpoints of the encapsulation and the endpoints for terminating the the transport protocol.

4.3. Encapsulator operation

Encapsulators create GUE data messages, set the source port to the

inner flow identifier, set flags and optional fields in the GUE header, and forward packets to a decapsulator.

An encapsulator may be an end host originating the packets of a flow, or may be a network device performing encapsulation on behalf of servers (routers implementing tunnels for instance). In either case, the intended target (decapsulator) is indicated by the outer destination IP address.

If an encapsulator is tunneling packets, that is encapsulating packets of layer 2 or layer 3 protocols (e.g. EtherIP, IPIP, ESP tunnel mode), it should follow standard conventions for tunneling of one IP protocol over another. Diffserv interaction with tunnels is described in [RFC2983], ECN propagation for tunnels is described in [RFC6040].

4.4. Decapsulator operation

A decapsulator performs decapsulation of GUE packets. A decapsulator is addressed by the outer destination IP address of a GUE packet. The decapsulator validates packets, including fields of the GUE header. If a packet is acceptable, the UDP and GUE headers are removed and the packet is resubmitted for IP protocol processing or control message processing if it is a control message.

If a decapsulator receives a GUE packet with an unsupported version, unknown flag, bad header length (too small for included optional fields), unknown control message type, or an otherwise malformed header, it must drop the packet and may log the event. No error message is returned back to the encapsulator. Note that set flags in GUE that are unknown to a decapsulator MUST NOT be ignored. If a GUE packet is received by a decapsulator with unknown flags, the packet MUST be dropped.

4.5. Router and switch operation

Routers and switches should forward GUE packets as standard UDP/IP packets. The outer five-tuple should contain sufficient information to perform flow classification corresponding to the flow of the inner packet. A switch should not normally need to parse a GUE header, and none of the flags or optional fields in the GUE header should affect routing.

A router should not modify a GUE header when forwarding a packet. It may encapsulate a GUE packet in another GUE packet, for instance to implement a network tunnel. In this case the router takes the role of an encapsulator, and the corresponding decapsulator is the logical endpoint of the tunnel.

4.6. Middlebox interactions

A middle box may interpret some flags and optional fields of the GUE header for classification purposes, but is not required to understand all flags and fields in GUE packets. A middle box should not drop a GUE packet because there are flags unknown to it. The header length in the GUE header allows a middlebox to inspect the payload packet without needing to parse the flags or optional fields.

A middlebox may infer bidirectional connection semantics to a UDP flow. For instance a stateful firewall may create a five-tuple rule to match flows on egress, and a corresponding five-tuple rule for matching ingress packets where the roles of source and destination are reversed for the IP addresses and UDP port numbers. To operate in this environment, a GUE tunnel must assume connected semantics defined by the UDP five tuple and the use of GUE encapsulation must be symmetric between both endpoints. The source port set in the UDP header must be the destination port the peer would set for replies.

4.7. NAT

IP address and port translation can be performed on the UDP/IP headers adhering to the requirements for NAT with UDP [RFC478]. In the case of stateful NAT, connection semantics must be applied to a GUE tunnel as described above.

When using transport mode encapsulation and traversing a NAT, the IP addresses may be changed such that the pseudo header checksum used for checksum calculation is modified and the checksum will be found invalid at the receiver. To compensate for this, A GUE option can be added which contains the checksum over the source and destination addresses when the packet is transmitted. Upon receiving this option, the delta of the pseudo header checksum is computed by subtracting the checksum over the source and and destination addresses from the checksum value in the option. The resultant value is then added into checksum calculation when validating the inner transport checksum.

4.8. Checksum Handling

This section describes the requirements around the UDP checksum and GUE header checksum. Checksums are an important consideration in that that they can provide end to end validation and protect against packet mis-delivery. The latter is allowed by the inclusion of a pseudo header that covers the IP addresses and UDP ports of the encapsulating headers.

4.8.1. Checksum requirements

The potential for mis-delivery of packets due to corruption of IP, UDP, or GUE headers must be considered. One of the following requirements must be met:

- o UDP checksums are enabled (for IPv4 or IPv6).
- o The GUE header checksum is used.
- o Zero UDP checksums are used in accordance with applicable requirements in [GREUDP], [RFC6935], and [RFC6936].

4.8.2. GUE header checksum

The GUE header checksum provides a UDP-lite [RFC3828] type of checksum capability as an optional field of the GUE header. The GUE header checksum minimally covers the GUE header and a GUE pseudo header. The GUE pseudo header includes the corresponding IP addresses as well as the UDP ports of the encapsulating headers. This checksum should provide adequate protection against address corruption in IPv6 when the UDP checksum is zero. Additionally, the GUE checksum provides protection of the GUE header when the UDP checksum is set to zero with either IPv4 or IPv6. The GUE header checksum is defined in [GUECSUM].

4.8.3. UDP Checksum with IPv4

For UDP in IPv4, the UDP checksum MUST be processed as specified in [RFC768] and [RFC1122] for both transmit and receive. An encapsulator MAY set the UDP checksum to zero for performance or implementation considerations. The IPv4 header includes a checksum that protects against mis-delivery of the packet due to corruption of IP addresses. The UDP checksum potentially provides protection against corruption of the UDP header, GUE header, and GUE payload. Enabling or disabling the use of checksums is a deployment consideration that should take into account the risk and effects of packet corruption, and whether the packets in the network are already adequately protected by other, possibly stronger mechanisms such as the Ethernet CRC. If an encapsulator sets a zero UDP checksum for IPv4 it SHOULD use the GUE header checksum as described in section 4.8.2.

When a decapsulator receives a packet, the UDP checksum field MUST be processed. If the UDP checksum is non-zero, the decapsulator MUST verify the checksum before accepting the packet. By default a decapsulator SHOULD accept UDP packets with a zero checksum. A node MAY be configured to disallow zero checksums per [RFC1122]; this may be done selectively, for instance disallowing zero checksums from certain hosts that are known to be sending over paths subject to

packet corruption. If verification of a non-zero checksum fails, a decapsulator lacks the capability to verify a non-zero checksum, or a packet with a zero-checksum was received and the decapsulator is configured to disallow, the packet MUST be dropped and an event MAY be logged.

4.8.4. UDP Checksum with IPv6

For UDP in IPv6, the UDP checksum MUST be processed as specified in [RFC768] and [RFC2460] for both transmit and receive. Unlike IPv4, there is no header checksum in IPv6 that protects against mis-delivery due to address corruption. Therefore, when GUE is used over IPv6, either the UDP checksum must be enabled or the GUE header checksum must be used. An encapsulator MAY set a zero UDP checksum for performance or implementation reasons, in which case the GUE header checksum MUST be used or applicable requirements for using zero UDP checksums in [GREUDP] MUST be met. If the UDP checksum is enabled, then the GUE header checksum should not be used since it is mostly redundant.

When a decapsulator receives a packet, the UDP checksum field MUST be processed. If the UDP checksum is non-zero, the decapsulator MUST verify the checksum before accepting the packet. By default a decapsulator MUST only accept UDP packets with a zero checksum if the GUE header checksum is used and is verified. If verification of a non-zero checksum fails, a decapsulator lacks the capability to verify a non-zero checksum, or a packet with a zero-checksum and no GUE header checksum was received, the packet MUST be dropped and an event MAY be logged.

4.9. MTU and fragmentation

Standard conventions for handling of MTU (Maximum Transmission Unit) and fragmentation in conjunction with networking tunnels (encapsulation of layer 2 or layer 3 packets) should be followed. Details are described in MTU and Fragmentation Issues with In-the-Network Tunneling [RFC4459]

If a packet is fragmented before encapsulation in GUE, all the related fragments must be encapsulated using the same source port (inner flow identifier). An operator may set MTU to account for encapsulation overhead and reduce the likelihood of fragmentation.

4.10. Congestion control

Per requirements of [RFC5405], if the IP traffic encapsulated with GUE implements proper congestion control no additional mechanisms should be required.

In the case that the encapsulated traffic does not implement any or sufficient control, or it is not known rather a transmitter will consistently implement proper congestion control, then congestion control at the encapsulation layer must be provided. Note this case applies to a significant use case in network virtualization in which guests run third party networking stacks that cannot be implicitly trusted to implement conformant congestion control.

Out of band mechanisms such as rate limiting, Managed Circuit Breaker, or traffic isolation may used to provide rudimentary congestion control. For finer grained congestion control that allows alternate congestion control algorithms, reaction time within an RTT, and interaction with ECN, in-band mechanisms may warranted.

DCCP may be used to provide congestion control for encapsulated flows. In this case, the protocol stack for an IP tunnel may be IP-GUE-DCCP-IP. Alternatively, GUE can be extended to include congestion control (related data carried in GUE optional fields). Congestion control mechanisms for GUE will be elaborated in other specifications.

4.11. Multicast

GUE packets may be multicast to decapsulators using a multicast destination address in the encapsulating IP headers. Each receiving host will decapsulate the packet independently following normal decapsulator operations. The receiving decapsulators should agree on the same set of GUE parameters and properties.

GUE allows encapsulation of unicast, broadcast, or multicast traffic. Entropy for the inner flow identifier (UDP source port) may be generated from the header of encapsulated unicast or broadcast/multicast packets at an encapsulator. The mapping mechanism between the encapsulated multicast traffic and the multicast capability in the IP network is transparent and independent to the encapsulation and is otherwise outside the scope of this document.

5. Inner flow identifier properties

5.1. Flow classification

A major objective of using GUE is that a network device can perform flow classification corresponding to the flow of the inner encapsulated packet based on the contents in the outer headers.

Hardware devices commonly perform hash computations on packet headers to classify packets into flows or flow buckets. Flow

classification is done to support load balancing (statistical multiplexing) of flows across a set of networking resources. Examples of such load balancing techniques are Equal Cost Multipath routing (ECMP), port selection in Link Aggregation, and NIC device Receive Side Scaling (RSS). Hashes are usually either a three-tuple hash of IP protocol, source address, and destination address; or a five-tuple hash consisting of IP protocol, source address, destination address, source port, and destination port. Typically, networking hardware will compute five-tuple hashes for TCP and UDP, but only three-tuple hashes for other IP protocols. Since the five-tuple hash provides more granularity, load balancing can be finer grained with better distribution. When a packet is encapsulated with GUE, the source port in the outer UDP packet is set to reflect the flow of the inner packet. When a device computes a five-tuple hash on the outer UDP/IP header of a GUE packet, the resultant value classifies the packet per its inner flow.

To support flow classification, the source port of the UDP header in GUE is set to a value that maps to the inner flow. This is referred to as the inner flow identifier. The inner flow identifier is set by the encapsulator; it can be computed on the fly based on packet contents or retrieved from a state maintained for the inner flow.

Examples of deriving an inner flow identifier are:

- o If the encapsulated packet is a layer 4 packet, TCP/IPv4 for instance, the inner flow identifier could be based on the canonical five-tuple hash of the inner packet.

- o If the encapsulated packet is an AH transport mode packet with TCP as next header, the inner flow identifier could be a hash over a three-tuple: TCP protocol and TCP ports of the encapsulated packet.

- o If a node is encrypting a packet using ESP tunnel mode and GUE encapsulation, the inner flow identifier could be based on the contents of clear-text packet. For instance, a canonical five-tuple hash for a TCP/IP packet could be used.

5.2. Inner flow identifier properties

The inner flow identifier is the value set in the UDP source port of a GUE packet. The inner flow identifier should adhere to the following properties:

- o The value set in the source port should be within the ephemeral port range. IANA suggests this range to be 49152 to 65535, where the high order two bits of the port are set to one. This

provides fourteen bits of entropy for the inner flow identifier.

- o The inner flow identifier should have a uniform distribution across encapsulated flows.

- o An encapsulator may occasionally change the inner flow identifier used for an inner flow per its discretion (for security, route selection, etc). Changing the value should happen no more than once every thirty seconds.

- o Decapsulators, or any networking devices, should not attempt any interpretation of the inner flow identifier, nor should they attempt to reproduce any hash calculation. They may use the value to match further receive packets for steering decisions, but cannot assume that the hash uniquely or permanently identifies a flow.

- o Input to the inner flow identifier is not restricted to ports and addresses; input could include flow label from an IPv6 packet, SPI from an ESP packet, or other flow related state in the encapsulator that is not necessarily conveyed in the packet.

- o The assignment function for inner flow identifiers should be randomly seeded to mitigate denial of service attacks. The seed may be changed periodically.

6. Motivation for GUE

This section presents the motivation for GUE with respect to other encapsulation methods.

A number of different encapsulation techniques have been proposed for the encapsulation of one protocol over another. EtherIP [RFC3378] provides layer 2 tunneling of Ethernet frames over IP. GRE [RFC2784], MPLS [RFC4023], and L2TP [RFC2661] provide methods for tunneling layer 2 and layer 3 packets over IP. NVGRE [NVGRE] and VXLAN [RFC7348] are proposals for encapsulation of layer 2 packets for network virtualization. IPIP [RFC2003] and Generic packet tunneling in IPv6 [RFC2473] provide methods for tunneling IP packets over IP.

Several proposals exist for encapsulating packets over UDP including ESP over UDP [RFC3948], TCP directly over UDP [TCPUDP], VXLAN, LISP [RFC6830] which encapsulates layer 3 packets, and Generic UDP Encapsulation for IP Tunneling (GRE over UDP)[GREUDP]. Generic UDP tunneling [GUT] is a proposal similar to GUE in that it aims to tunnel packets of IP protocols over UDP.

GUE has the following discriminating features:

- o UDP encapsulation leverages specialized network device processing for efficient transport. The semantics for using the UDP source port as an identifier for an inner flow are defined.

- o GUE permits encapsulation of arbitrary IP protocols, which includes layer 2 3, and 4 protocols. This potentially allows nearly all traffic within a data center to be normalized to be either TCP or UDP on the wire.

- o Multiple protocols can be multiplexed over a single UDP port number. This is in contrast to techniques to encapsulate protocols over UDP using a protocol specific port number (such as ESP/UDP, GRE/UDP, SCTP/UDP). GUE provides a uniform and extensible mechanism for encapsulating all IP protocols in UDP with minimal overhead (four bytes of additional header).

- o GUE is extensible. New flags and optional fields can be defined.

- o The GUE header includes a header length field. This allows a network node to inspect an encapsulated packet without needing to parse the full encapsulation header.

- o Private data in the encapsulation header allows local customization and experimentation while being compatible with processing in network nodes (routers and middleboxes).

- o GUE includes both data messages (encapsulation of packets) and control messages (such as OAM).

7. Security Considerations

Encapsulation of IP protocols within GUE should not increase security risk, nor provide additional security in itself. As suggested in section 5 the source port for of UDP packets in GUE should be randomly seeded to mitigate some possible denial service attacks.

Security for Generic UDP Encapsulation, including security for the GUE header and payload, is described in detail in [GUESEC].

8. IANA Consideration

A user UDP port number assignment for GUE has been assigned:

```
Service Name: gue
Transport Protocol(s): UDP
Assignee: Tom Herbert <therbert@google.com>
Contact: Tom Herbert <therbert@google.com>
Description: Generic UDP Encapsulation
```

Reference: draft-herbert-gue
Port Number: 6080
Service Code: N/A
Known Unauthorized Uses: N/A
Assignment Notes: N/A

IANA is requested to create a "GUE flag-fields" registry to allocate flags and optional fields for the primary GUE header flags and extension flags. This shall be a registry of bit assignments for flags, length of optional fields for corresponding flags, and descriptive strings. There are sixteen bits for primary GUE header flags (bit number 0-15) where bit 15 is reserved as the extension flag in this document. There are thirty-two bits for extension flags.

9. Acknowledgements

The authors would like to thank David Liu, Erik Nordmark, and Fred Templin for valuable input on this draft.

10. References

10.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, DOI 10.17487/RFC2434, October 1998, <http://www.rfc-editor.org/info/rfc2434>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <http://www.rfc-editor.org/info/rfc2983>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <http://www.rfc-editor.org/info/rfc6040>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <http://www.rfc-editor.org/info/rfc6936>.

[RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", RFC 4459, DOI 10.17487/RFC4459, April

2006, <http://www.rfc-editor.org/info/rfc4459>.

10.2. Informative References

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <http://www.rfc-editor.org/info/rfc2003>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <http://www.rfc-editor.org/info/rfc3948>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <http://www.rfc-editor.org/info/rfc6830>.

[RFC3378] Housley, R. and S. Hollenbeck, "EtherIP: Tunneling Ethernet Frames in IP Datagrams", RFC 3378, DOI 10.17487/RFC3378, September 2002, <http://www.rfc-editor.org/info/rfc3378>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <http://www.rfc-editor.org/info/rfc2784>.

[RFC4023] Worster, T., Rekhter, Y., and E. Rosen, Ed., "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005, <http://www.rfc-editor.org/info/rfc4023>.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, DOI 10.17487/RFC2661, August 1999, <http://www.rfc-editor.org/info/rfc2661>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., Ed., and G. Fairhurst, Ed., "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004, <http://www.rfc-editor.org/info/rfc3828>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

[NVGRE] Garg, P., and Wang, Y., "NVGRE: Network Virtualization using Generic Routing Encapsulation" draft-sridharan-virtualization-nvgre-08

[TCPUDP] Chesire, S., Graessley, J., and McGuire, R., "Encapsulation of TCP and other Transport Protocols over UDP" draft-cheshire-tcp-over-udp-00

[GREUDP] Crabbe, E., Yong, L., Xu, X., and Herbert, T.,Cpsest "Generic UDP Encapsulation for IP Tunneling" draft-ietf-tsvwg-gre-in-udp-encap-06

[GUESEC] Yong, L., Herbert, T., "Generic UDP Encapsulation (GUE) for Secure Transport", draft-hy-gue-4-secure-transport-02, work in progress.

[GUT] Manner, J., Varia, N., and Briscoe, B., "Generic UDP Tunnelling (GUT) draft-manner-tsvwg-gut-02.txt"

[REMCSUM] Herbert, T., "Remote Checksum Offload" draft-herbert-remotecsumoffload-00

Appendix A: NIC processing for GUE

This appendix provides some guidelines for Network Interface Cards (NICs) to implement common offloads and accelerations to support GUE. Note that most of this discussion is generally applicable to other methods of UDP based encapsulation.

A.1. Receive multi-queue

Contemporary NICs support multiple receive descriptor queues (multi-queue). Multi-queue enables load balancing of network processing for a NIC across multiple CPUs. On packet reception, a NIC must select the appropriate queue for host processing. Receive Side Scaling is a common method which uses the flow hash for a packet to index an indirection table where each entry stores a queue number. Flow Director and Accelerated Receive Flow Steering (aRFS) allow a host to program the queue that is used for a given flow which is identified either by an explicit five-tuple or by the flow’s hash.

GUE encapsulation should be compatible with multi-queue NICs that support five-tuple hash calculation for UDP/IP packets as input to RSS. The inner flow identifier (source port) ensures classification of the encapsulated flow even in the case that the outer source and destination addresses are the same for all flows (e.g. all flows are going over a single tunnel).

By default, UDP RSS support is often disabled in NICs to avoid out of order reception that can occur when UDP packets are fragmented. As discussed above, fragmentation of GUE packets should be mitigated by fragmenting packets before entering a tunnel, path MTU discovery in higher layer protocols, or operator adjusting MTUs. Other UDP traffic may not implement such procedures to avoid fragmentation, so enabling UDP RSS support in the NIC should be a considered tradeoff during configuration.

A.2. Checksum offload

Many NICs provide capabilities to calculate standard ones complement payload checksum for packets in transmit or receive. When using GUE encapsulation there are at least two checksums that may be of interest: the encapsulated packet’s transport checksum, and the UDP checksum in the outer header.

A.2.1. Transmit checksum offload

NICs may provide a protocol agnostic method to offload transmit checksum (NETIF_F_HW_CSUM in Linux parlance) that can be used with GUE. In this method the host provides checksum related parameters in a transmit descriptor for a packet. These parameters include the starting offset of data to checksum, the length of data to checksum, and the offset in the packet where the computed checksum is to be written. The host initializes the checksum field to pseudo header checksum.

In the case of GUE, the checksum for an encapsulated transport layer packet, a TCP packet for instance, can be offloaded by setting the appropriate checksum parameters.

NICs typically can offload only one transmit checksum per packet, so simultaneously offloading both an inner transport packet’s checksum and the outer UDP checksum is likely not possible. In this case setting UDP checksum to zero (per above discussion) and offloading the inner transport packet checksum might be acceptable.

If an encapsulator is co-resident with a host, then checksum offload may be performed using remote checksum offload [REMCSUM]. Remote checksum offload relies on NIC offload of the simple UDP/IP checksum

which is commonly supported even in legacy devices. In remote checksum offload the outer UDP checksum is set and the GUE header includes an option indicating the start and offset of the inner "offloaded" checksum. The inner checksum is initialized to the pseudo header checksum. When a decapsulator receives a GUE packet with the remote checksum offload option, it completes the offload operation by determining the packet checksum from the indicated start point to the end of the packet, and then adds this into the checksum field at the offset given in the option. Computing the checksum from the start to end of packet is efficient if checksum-complete is provided on the receiver.

A.2.2. Receive checksum offload

GUE is compatible with NICs that perform a protocol agnostic receive checksum (CHECKSUM_COMPLETE in Linux parlance). In this technique, a NIC computes a ones complement checksum over all (or some predefined portion) of a packet. The computed value is provided to the host stack in the packet’s receive descriptor. The host driver can use this checksum to "patch up" and validate any inner packet transport checksum, as well as the outer UDP checksum if it is non-zero.

Many legacy NICs don’t provide checksum-complete but instead provide an indication that a checksum has been verified (CHECKSUM_UNNECESSARY in Linux). Usually, such validation is only done for simple TCP/IP or UDP/IP packets. If a NIC indicates that a UDP checksum is valid, the checksum-complete value for the UDP packet is the "not" of the pseudo header checksum. In this way, checksum-unnecessary can be converted to checksum-complete. So if the NIC provides checksum-unnecessary for the outer UDP header in an encapsulation, checksum conversion can be done so that the checksum-complete value is derived and can be used by the stack to validate an checksums in the encapsulated packet.

A.3. Transmit Segmentation Offload

Transmit Segmentation Offload (TSO) is a NIC feature where a host provides a large (>MTU size) TCP packet to the NIC, which in turn splits the packet into separate segments and transmits each one. This is useful to reduce CPU load on the host.

The process of TSO can be generalized as:

- Split the TCP payload into segments which allow packets with size less than or equal to MTU.

- For each created segment:

  1. Replicate the TCP header and all preceding headers of the

original packet.

2. Set payload length fields in any headers to reflect the length of the segment.

3. Set TCP sequence number to correctly reflect the offset of the TCP data in the stream.

4. Recompute and set any checksums that either cover the payload of the packet or cover header which was changed by setting a payload length.

Following this general process, TSO can be extended to support TCP encapsulation in GUE. For each segment the Ethernet, outer IP, UDP header, GUE header, inner IP header if tunneling, and TCP headers are replicated. Any packet length header fields need to be set properly (including the length in the outer UDP header), and checksums need to be set correctly (including the outer UDP checksum if being used).

To facilitate TSO with GUE it is recommended that optional fields should not contain values that must be updated on a per segment basis-- for example the GUE fields should not include checksums, lengths, or sequence numbers that refer to the payload. If the GUE header does not contain such fields then the TSO engine only needs to copy the bits in the GUE header when creating each segment and does not need to parse the GUE header.

A.4. Large Receive Offload

Large Receive Offload (LRO) is a NIC feature where packets of a TCP connection are reassembled, or coalesced, in the NIC and delivered to the host as one large packet. This feature can reduce CPU utilization in the host.

LRO requires significant protocol awareness to be implemented correctly and is difficult to generalize. Packets in the same flow need to be unambiguously identified. In the presence of tunnels or network virtualization, this may require more than a five-tuple match (for instance packets for flows in two different virtual networks may have identical five-tuples). Additionally, a NIC needs to perform validation over packets that are being coalesced, and needs to fabricate a single meaningful header from all the coalesced packets.

The conservative approach to supporting LRO for GUE would be to assign packets to the same flow only if they have identical five-tuple and were encapsulated the same way. That is the outer IP addresses, the outer UDP ports, GUE protocol, GUE flags and fields, and inner five tuple are all identical.

Appendix B: Privileged ports

Using the source port to contain an inner flow identifier value disallows the security method of a receiver enforcing that the source port be a privileged port. Privileged ports are defined by some operating systems to restrict source port binding. Unix, for instance, considered port number less than 1024 to be privileged.

Enforcing that packets are sent from a privileged port is widely considered an inadequate security mechanism and has been mostly deprecated. To approximate this behavior, an implementation could restrict a user from sending a packet destined to the GUE port without proper credentials.

Appendix C: Inner flow identifier as a route selector

An encapsulator generating an inner flow identifier may modulate the value to perform a type of multipath source routing. Assuming that networking switches perform ECMP based on the flow hash, a sender can affect the path by altering the inner flow identifier. For instance, a host may store a flow hash in its PCB for an inner flow, and may alter the value upon detecting that packets are traversing a lossy path. Changing the inner flow identifier for a flow should be subject to hysteresis (at most once every thirty seconds) to limit the number of out of order packets.

Appendix D: Hardware protocol implementation considerations

A low level protocol, such is GUE, is likely interesting to being supported by high speed network devices. Variable length header (VLH) protocols like GUE are often considered difficult to efficiently implement in hardware. In order to retain the important characteristics of an extensible and robust protocol, hardware vendors may practice "constrained flexibility". In this model, only certain combinations or protocol header parameterizations are implemented in hardware fast path. Each such parameterization is fixed length so that the particular instance can be optimized as a fixed length protocol. In the case of GUE this constitutes specific combinations of GUE flags, fields, and next protocol. The selected combinations would naturally be the most common cases which form the "fast path", and other combinations are assumed to take the "slow path".

In time, needs and requirements of the protocol may change which may manifest themselves as new parameterizations to be supported in the fast path. To allow allow this extensibility, a device practicing constrained flexibility should allow the fast path parameterizations to be programmable.

Authors' Addresses

Tom Herbert
Facebook
1 Hacker Way
Menlo Park, CA 94052
US

Email: tom@herbertland.com

Lucy Yong
Huawei USA
5340 Legacy Dr.
Plano, TX 75024
US

Email: lucy.yong@huawei.com

Osama Zia
Microsoft
1 Microsoft Way
Redmond, WA 98029
US

Email: osamaz@microsoft.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 2, 2015

P. Quinn
Cisco Systems, Inc.
R. Manur
Broadcom
L. Kreeger
D. Lewis
F. Maino
M. Smith
Cisco Systems, Inc.
P. Agarwal
Innovium, Inc
L. Yong
Huawei USA
X. Xu
Huawei Technologies
U. Elzur
Intel
P. Garg
Microsoft
D. Melman
Marvell

May 1, 2015

Generic Protocol Extension for VXLAN
draft-ietf-nvo3-vxlan-gpe-00.txt

Abstract

This draft describes extending Virtual eXtensible Local Area Network (VXLAN), via changes to the VXLAN header, with three new capabilities: support for multi-protocol encapsulation, operations, administration and management (OAM) signaling and explicit versioning.

Status of this Memo

material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 2, 2015.

Copyright Notice

Table of Contents

1. Introduction

Virtual eXtensible Local Area Network VXLAN [RFC7348] defines an encapsulation format that encapsulates Ethernet frames in an outer UDP/IP transport. As data centers evolve, the need to carry other protocols encapsulated in an IP packet is required, as well as the need to provide increased visibility and diagnostic capabilities within the overlay. The VXLAN header does not specify the protocol being encapsulated and therefore is currently limited to encapsulating only Ethernet frame payload, nor does it provide the ability to define OAM protocols. In addition, [RFC6335] requires that new transports not use transport layer port numbers to identify tunnel payload, rather it encourages encapsulations to use their own identifiers for this purpose. VXLAN GPE is intended to extend the existing VXLAN protocol to provide protocol typing, OAM, and versioning capabilities.

The Version and OAM bits are introduced in Section 3, and the choice of location for these fields is driven by minimizing the impact on existing deployed hardware.

In order to facilitate deployments of VXLAN GPE with hardware currently deployed to support VXLAN, changes from legacy VXLAN have been kept to a minimum. Section 5 provides a detailed discussion about how VXLAN GPE addresses the requirement for backward compatibility with VXLAN.

2. VXLAN Without Protocol Extension

VXLAN provides a method of creating multi-tenant overlay networks by encapsulating packets in IP/UDP along with a header containing a network identifier which is used to isolate tenant traffic in each overlay network from each other. This allows the overlay networks to run over an existing IP network.

Through this encapsulation, VXLAN creates stateless tunnels between VXLAN Tunnel End Points (VTEPs) which are responsible for adding/removing the IP/UDP/VXLAN headers and providing tenant traffic isolation based on the VXLAN Network Identifier (VNI). Tenant systems are unaware that their networking service is being provided by an overlay.

When encapsulating packets, a VTEP must know the IP address of the proper remote VTEP at the far end of the tunnel that can deliver the inner packet to the Tenant System corresponding to the inner destination address. In the case of tenant multicast or broadcast, the outer IP address may be an IP multicast group address, or the VTEP may replicate the packet and send it to all known VTEPs. If multicast is used in the underlay network to send encapsulated packets to remote VTEPs, Any Source Multicast is used and each VTEP serving a particular VNI must perform a (*, G) join to the same group IP address.

Inner to outer address mapping can be determined in two ways. One is source based learning in the data plane, and the other is distribution via a control plane.

Source based learning requires a receiving VTEP to create an inner to outer address mapping by gleaning the information from the received packets by correlating the inner source address to the outer source IP address. When a mapping does not exist, a VTEP forwards the packets to all remote VTEPs participating in the VNI by using IP multicast in the IP underlay network. Each VTEP must be configured with the IP multicast address to use for each VNI. How this occurs is out of scope.

The control plane used to distribute inner to outer mappings is also out of scope. It could use a centralized authority or be distributed, or use a hybrid.

The VXLAN Network Identifier (VNI) provides scoping for the addresses in the header of the encapsulated PDU. If the encapsulated packet is an Ethernet frame, this means the Ethernet MAC addresses are only unique within a given VNI and may overlap with MAC addresses within a different VNI. If the encapsulated packet is an IP packet, this

means the IP addresses are only unique within that VNI.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|R|R|I|R|R|R|            Reserved                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: VXLAN Header

3. Generic Protocol Extension for VXLAN (VXLAN GPE)

3.1. VXLAN GPE Header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|Ver|I|P|R|O|       Reserved                |Next Protocol  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: VXLAN GPE Header

Flags (8 bits): The first 8 bits of the header are the flag field. The bits designated "R" above are reserved flags. These MUST be set to zero on transmission and ignored on receipt.

Version (Ver): Indicates VXLAN GPE protocol version. The initial version is 0. If a receiver does not support the version indicated it MUST drop the packet.

Instance Bit (I bit): The I bit MUST be set to indicate a valid VNI.

Next Protocol Bit (P bit): The P bit is set to indicate that the Next Protocol field is present.

OAM Flag Bit (O bit): The O bit is set to indicate that the packet is an OAM packet.

Next Protocol: This 8 bit field indicates the protocol header immediately following the VXLAN GPE header.

VNI: This 24 bit field identifies the VXLAN overlay network the inner packet belongs to. Inner packets belonging to different VNIs cannot communicate with each other (unless explicitly allowed by policy).

Reserved: Reserved fields MUST be set to zero on transmission and ignored on receipt.

3.2. Multi Protocol Support

This draft defines the following two changes to the VXLAN header in order to support multi-protocol encapsulation:

P Bit: Flag bit 5 is defined as the Next Protocol bit. The P bit MUST be set to 1 to indicate the presence of the 8 bit next protocol field. When P=1, the destination UDP port MUST be 4790.

P = 0 indicates that the payload MUST conform to VXLAN as defined in [RFC7348], including destination UDP port.

Flag bit 5 was chosen as the P bit because this flag bit is currently reserved in VXLAN.

Next Protocol Field: The lower 8 bits of the first word are used to carry a next protocol. This next protocol field contains the protocol of the encapsulated payload packet. A new protocol registry will be requested from IANA, see section 9.2.

This draft defines the following Next Protocol values:

0x1 : IPv4
0x2 : IPv6
0x3 : Ethernet
0x4 : Network Service Header [NSH]

3.3. OAM Support

Flag bit 7 is defined as the O bit. When the O bit is set to 1, the packet is an OAM packet and OAM processing MUST occur. Other header fields including Next Protocol MUST adhere to the definitions in section 3. The OAM protocol details are out of scope for this document. As with the P-bit, bit 7 is currently a reserved flag in VXLAN.

3.4. Version Bits

VXLAN GPE bits 2 and 3 are defined as version bits. These bits are reserved in VXLAN. The version field is used to ensure backward compatibility going forward with future VXLAN GPE updates.

The initial version for VXLAN GPE is 0.

4. Outer Encapsulations

In addition to the VXLAN GPE header, the packet is further encapsulated in UDP and IP. Data centers based on Ethernet, will then send this IP packet over Ethernet.

Outer UDP Header:

Destination UDP Port: IANA has assigned the value 4790 for the VXLAN GPE UDP port. This well-known destination port is used when sending VXLAN GPE encapsulated packets.

Source UDP Port: The source UDP port is used as entropy for devices forwarding encapsulated packets across the underlay (ECMP for IP routers, or load splitting for link aggregation by bridges). Tenant traffic flows should all use the same source UDP port to lower the chances of packet reordering by the underlay for a given flow. It is recommended for VTEPs to generate this port number using a hash of the inner packet headers.

UDP Checksum: Source VTEPs MAY either calculate a valid checksum, or if this is not possible, set the checksum to zero. When calculating a checksum, it MUST be calculated across the entire packet (outer IP header, UDP header, VXLAN GPE header and payload packet). All receiving VTEPs must accept a checksum value of zero. If the receiving VTEP is capable of validating the checksum, it MAY validate a non-zero checksum and MUST discard the packet if the checksum is determined to be invalid.

Outer IP Header:

This is the header used by the underlay network to deliver packets between VTEPs. The destination IP address can be a unicast or a multicast IP address. The source IP address must be the source VTEP IP address which can be used to return tenant packets to the tenant system source address within the inner packet header.

When the outer IP header is IPv4, VTEPs MUST set the DF bit.

Outer Ethernet Header:

Most data centers networks are built on Ethernet. Assuming the outer IP packet is being sent across Ethernet, there will be an Ethernet header used to deliver the IP packet to the next hop, which could be the destination VTEP or be a router used to forward the IP packet towards the destination VTEP. If VLANs are in use within the data center, then this Ethernet header would also contain a VLAN tag.

The following figures show the entire stack of protocol headers that would be seen on an Ethernet link carrying encapsulated packets from a VTEP across the underlay network for both IPv4 and IPv6 based underlay networks.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Outer Ethernet Header:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Outer Destination MAC Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Outer Destination MAC Address | Outer Source MAC Address      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Outer Source MAC Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt Ethertype = C-Tag 802.1Q  |     Outer VLAN Tag            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Ethertype = 0x0800            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Outer IPv4 Header:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |Protocl=17(UDP)|   Header Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Outer Source IPv4 Address               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Outer Destination IPv4 Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Outer UDP Header:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Source Port         |       Dest Port = 4790        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           UDP Length          |        UDP Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

VXLAN GPE Header:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
|R|R|Ver|I|P|R|O|       Reserved                |Next Protocol  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Payload:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Depends on VXLAN GPE Next Protocol field above.           |
|    Note that if the payload is Ethernet, then the original    |
|    Ethernet Frame's FCS is not included.                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Frame Check Sequence:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   New FCS (Frame Check Sequence) for Outer Ethernet Frame     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Outer Headers for VXLAN GPE over IPv4

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Outer Ethernet Header:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Outer Destination MAC Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Outer Destination MAC Address | Outer Source MAC Address      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Outer Source MAC Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt Ethertype = C-Tag 802.1Q  |     Outer VLAN Tag            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Ethertype = 0x86DD            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Outer IPv6 Header:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        | NxtHdr=17(UDP)|   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
```

```
      |                                                               |
      +                     Outer Source IPv6 Address                 +
      |                                                               |
      +                                                               +
      |                                                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                                                               |
      +                                                               +
      |                                                               |
      +                   Outer Destination IPv6 Address              +
      |                                                               |
      +                                                               +
      |                                                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

      Outer UDP Header:
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |           Source Port         |       Dest Port = 4790        |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |           UDP Length          |        UDP Checksum           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

      VXLAN GPE Header:
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |R|R|Ver|I|P|R|O|       Reserved                |Next Protocol  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                VXLAN Network Identifier (VNI) |   Reserved    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

      Payload:
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |      Depends on VXLAN GPE Next Protocol field above.          |
      |    Note that if the payload is Ethernet, then the original    |
      |    Ethernet Frame’s FCS is not included.                      |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

      Frame Check Sequence:
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |   New FCS (Frame Check Sequence) for Outer Ethernet Frame     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure X: Outer Headers for VXLAN GPE over IPv6

Figure 4: Outer Headers for VXLAN GPE over IPv6

4.1. Inner VLAN Tag Handling

If the inner packet (as indicated by the VXLAN GPE Next Protocol field) is an Ethernet frame, it is recommended that it does not contain a VLAN tag. In the most common scenarios, the tenant VLAN tag is translated into a VXLAN Network Identifier. In these scenarios, VTEPs should never send an inner Ethernet frame with a VLAN tag, and a VTEP performing decapsulation should discard any inner frames received with a VLAN tag. However, if the VTEPs are specifically configured to support it for a specific VXLAN Network Identifier, a VTEP may support transparent transport of the inner VLAN tag between all tenant systems on that VNI. The VTEP never looks at the value of the inner VLAN tag, but simply passes it across the underlay.

4.2. Fragmentation Considerations

VTEPs MUST never fragment an encapsulated VXLAN GPE packet, and when the outer IP header is IPv4, VTEPs MUST set the DF bit in the outer IPv4 header. It is recommended that the underlay network be configured to carry an MTU at least large enough to accommodate the added encapsulation headers. It is recommended that VTEPs perform Path MTU discovery [RFC1191] [RFC1981] to determine if the underlay network can carry the encapsulated payload packet.

5. Backward Compatibility

5.1. VXLAN VTEP to VXLAN GPE VTEP

A VXLAN VTEP conforms to VXLAN frame format and uses UDP destination port 4789 when sending traffic to VXLAN GPE VTEP. As per VXLAN, reserved bits 5 and 7, VXLAN GPE P and O-bits respectively must be set to zero. The remaining reserved bits must be zero, including the VXLAN GPE version field, bits 2 and 3. The encapsulated payload MUST be Ethernet.

5.2. VXLAN GPE VTEP to VXLAN VTEP

A VXLAN GPE VTEP MUST NOT encapsulate non-Ethernet frames to a VXLAN VTEP. When encapsulating Ethernet frames to a VXLAN VTEP, the VXLAN GPE VTEP MUST conform to VXLAN frame format and hence will set the P bit to 0, the Next Protocol to 0 and use UDP destination port 4789. A VXLAN GPE VTEP MUST also set O = 0 and Ver = 0 when encapsulating Ethernet frames to VXLAN VTEP. The receiving VXLAN VTEP will treat this packet as a VXLAN packet.

A method for determining the capabilities of a VXLAN VTEP (GPE or non-GPE) is out of the scope of this draft.

5.3. VXLAN GPE UDP Ports

VXLAN GPE uses a IANA assigned UDP destination port, 4790, when sending traffic to VXLAN GPE VTEPs.

5.4. VXLAN GPE and Encapsulated IP Header Fields

When encapsulating and decapsulating IPv4 and IPv6 packets, certain fields, such as IPv4 Time to Live (TTL) from the inner IP header need to be considered. VXLAN GPE IP encapsulation and decapsulation utilizes the techniques described in [RFC6830], section 5.3.

6. VXLAN GPE Examples

This section provides three examples of protocols encapsulated using the Generic Protocol Extension for VXLAN described in this document.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|0|0|I|1|R|0|       Reserved                |    NP = IPv4  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Original IPv4 Packet                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: IPv4 and VXLAN GPE

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|0|0|I|1|R|0|       Reserved                |  NP = IPv6    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Original IPv6 Packet                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: IPv6 and VXLAN GPE

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|0|0|I|1|R|0|       Reserved                |NP = Ethernet  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Original Ethernet Frame                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Ethernet and VXLAN GPE

7. Security Considerations

VXLAN’s security is focused on issues around L2 encapsulation into L3. With VXLAN GPE, issues such as spoofing, flooding, and traffic redirection are dependent on the particular protocol payload encapsulated.

8. Acknowledgments

A special thank you goes to Dino Farinacci for his guidance and detailed review.

9. IANA Considerations

9.1. UDP Port

UDP 4790 port has been assigned by IANA for VXLAN GPE.

9.2. VXLAN GPE Next Protocol

IANA is requested to set up a registry of "Next Protocol". These are 8-bit values. Next Protocol values 0, 1, 2, 3 and 4 are defined in this draft. New values are assigned via Standards Action [RFC5226].

Next Protocol	Description	Reference
0	Reserved	This document
1	IPv4	This document
2	IPv6	This document
3	Ethernet	This document
4	NSH	This document
5..253	Unassigned	

Table 1

9.3. VXLAN GPE Flag and Reserved Bits

There are ten flag bits at the beginning of the VXLAN GPE header, followed by 16 reserved bits and an 8-bit reserved field at the end of the header. New bits are assigned via Standards Action [RFC5226].

Bits 0-1 - Reserved
Bits 2-3 - Version
Bit 4 - Instance ID (I bit)
Bit 5 - Next Protocol (P bit)
Bit 6 - Reserved
Bit 7 - OAM (O bit)
Bits 8-23 - Reserved
Bits 24-31 in the 2nd Word -- Reserved

Reserved bits/fields MUST be set to 0 by the sender and ignored by the receiver.

10. References

10.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

10.2. Informative References

[NSH] Quinn, P., Ed. and U. Elzur, Ed., "Network Service Header", 2015.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.

Authors' Addresses

Paul Quinn
Cisco Systems, Inc.

Email: paulq@cisco.com

Rajeev Manur
Broadcom

Email: rmanur@broadcom.com

Larry Kreeger
Cisco Systems, Inc.

Email: kreeger@cisco.com

Darrel Lewis
Cisco Systems, Inc.

Email: darlewis@cisco.com

Fabio Maino
Cisco Systems, Inc.

Email: fmaino@cisco.com

Michael Smith
Cisco Systems, Inc.

Email: michsmit@cisco.com

Puneet Agarwal
Innovium, Inc

Email: puneet@acm.org

Lucy Yong
Huawei USA

Email: lucy.yong@huawei.com

Xiaohu Xu
Huawei Technologies

Email: xuxiaohu@huawei.com

Uri Elzur
Intel

Email: uri.elzur@intel.com

Pankaj Garg
Microsoft

Email: Garg.Pankaj@microsoft.com

David Melman
Marvell

Email: davidme@marvell.com

RTGWG
Internet-Draft
Intended status: Informational
Expires: December 3, 2015

E. Nordmark (ed)
Arista Networks
A. Tian
Ericsson Inc.
J. Gross
VMware
J. Hudson
Brocade Communications Systems, Inc.
L. Kreeger
Cisco Systems, Inc.
P. Garg
Microsoft
P. Thaler
Broadcom Corporation
T. Herbert
Facebook
June 2015

Encapsulation Considerations
draft-ietf-rtgwg-dt-encap-00

Abstract

The IETF Routing Area director has chartered a design team to look at common issues for the different data plane encapsulations being discussed in the NVO3 and SFC working groups and also in the BIER BoF, and also to look at the relationship between such encapsulations in the case that they might be used at the same time. The purpose of this design team is to discover, discuss and document considerations across the different encapsulations in the different WGs/BoFs so that we can reduce the number of wheels that need to be reinvented in the future.

Status of this Memo

material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 3, 2015.

Copyright Notice

Table of Contents

1. Design Team Charter

There have been multiple efforts over the years that have resulted in new or modified data plane behaviors involving encapsulations. That includes IETF efforts like MPLS, LISP, and TRILL but also industry efforts like VXLAN and NVGRE. These collectively can be seen as a source of insight into the properties that data planes need to meet. The IETF is currently working on potentially new encapsulations in NVO3 and SFC and considering working on BIER. In addition there is work on tunneling in the INT area.

This is a short term design team chartered to collect and construct useful advice to parties working on new or modified data plane behaviors that include additional encapsulations. The goal is for the group to document useful advice gathered from interacting with ongoing efforts. An Internet Draft will be produced for IETF92 to capture that advice, which will be discussed in RTGWG.

Data plane encapsulations face a set of common issues such as:

- o How to provide entropy for ECMP
- o Issues around packet size and fragmentation/reassembly
- o OAM - what support is needed in an encapsulation format?
- o Security and privacy.
- o QoS
- o Congestion Considerations
- o IPv6 header protection (zero UDP checksum over IPv6 issue)
- o Extensibility - e.g., for evolving OAM, security, and/or congestion control
- o Layering of multiple encapsulations e.g., SFC over NVO3 over BIER

The design team will provide advice on those issues. The intention is that even where we have different encapsulations for different purposes carrying different information, each such encapsulation doesn’t have to reinvent the wheel for the above common issues.

The design team will look across the routing area in particular at SFC, NVO3 and BIER. It will not be involved in comparing or analyzing any particular encapsulation formats proposed in those WGs and BoFs but instead focus on common advice.

2. Overview

The references provide background information on NVO3, SFC, and BIER. In particular, NVO3 is introduced in [RFC7364], [RFC7365], and [I-D.ietf-nvo3-arch]. SFC is introduced in [I-D.ietf-sfc-architecture] and [I-D.ietf-sfc-problem-statement]. Finally, the information on BIER is in [I-D.shepherd-bier-problem-statement],

[I-D.wijnands-bier-architecture], and [I-D.wijnands-mpls-bier-encapsulation]. We assume the reader has some basic familiarity with those proposed encapsulations. The Related Work section points at some prior work that relates to the encapsulation considerations in this document.

Encapsulation protocols typically have some unique information that they need to carry. In some cases that information might be modified along the path and in other cases it is constant. The in-flight modifications has impacts on what it means to provide security for the encapsulation headers.

- o NVO3 carries a VNI Identifier edge to edge which is not modified. There has been OAM discussions in the WG and it isn’t clear whether some of the OAM information might be modified in flight.
- o SFC carries Service Function Path identification and service meta-data. The meta-data might be modified as the packets follow the service path. SFC talks of some loop avoidance mechanism which is likely to result in modifications for for each hop in the service chain even if the meta-data is unmodified.
- o BIER carries a bitmap of egress ports to which a packet should be delivered, and as the packet is forwarded down different paths different bits are cleared in that bitmap.

Even if information isn’t modified in flight there might be devices that wish to inspect that information. For instance, one can envision future NVO3 security devices which filter based on the virtual network identifier.

The need for extensibility is different across the protocols

- o NVO3 might need some extensions for OAM and security.
- o SFC consists of Service Function Path identification plus carrying service meta-data along a path, and different services might need different types and amount of meta-data.
- o BIER might need variable number of bits in their bitmaps, or other future schemes to scale up to larger network.

The extensibility needs and constraints might be different when considering hardware vs. software implementations of the encapsulation headers. NIC hardware might have different constraints than switch hardware.

As the IETF designs these encapsulations the different WGs solve the issues for their own encapsulation. But there are likely to be future cases when the different encapsulations are combined in the same header. For instance, NVO3 might be a "transport" used to carry SFC between the different hops in the service chain.

Most of the issues discussed in this document are not new. The IETF and industry as specified and deployed many different encapsulation

or tunneling protocols over time, ranging from simple IP-in-IP and GRE encapsulation, IPsec, pseudo-wires, session-based approached like L2TP, and the use of MPLS control and data planes. IEEE 802 has also defined layered encapsulation for Provider Backbone Bridges (PBB) and IEEE 802.1Qbp (ECMP). This document tries to leverage what we collectively have learned from that experience and summarize what would be relevant for new encapsulations like NVO3, SFC, and BIER.

3. Common Issues

[This section is mostly a repeat of the charter but with a few modifications and additions.]

Any new encapsulation protocol would need to address a large set of issues that are not central to the new information that this protocol intends to carry. The common issues explored in this document are:

- o How to provide entropy for Equal Cost MultiPath (ECMP) routing
- o Issues around packet size and fragmentation/reassembly
- o Next header indication - each encapsulation might be able to carry different payloads
- o OAM - what support is needed in an encapsulation format?
- o Security and privacy
- o QoS
- o Congestion Considerations
- o Header protection
- o Extensibility - e.g., for evolving OAM, security, and/or congestion control
- o Layering of multiple encapsulations e.g., SFC over NVO3 over BIER
- o Importance of being friendly to hardware and software implementations

The degree to which these common issues apply to a particular encapsulation can differ based on the intended purpose of the encapsulation. But it is useful to understand all of them before determining which ones apply.

4. Scope

It is important to keep in mind what we are trying to cover and not cover in this document and effort. This is

- o A look across the three new encapsulations, while taking lots of previous work into account
- o Focus on the class of encapsulations that would run over IP/UDP. That was done to avoid being distracted by the data-plane and control-plane interaction, which is more significant for protocols that are designed to run over "transports" that maintain session

or path state.
- o We later expanded the scope somewhat to consider how the encapsulations would play with MPLS "transport", which is important because SFC and BIER seem to target being independent of the underlying "transport"

However, this document and effort is NOT intended to:
- o Design some new encapsulation header to rule them all
- o Design yet another new NVO3 encapsulation header
- o Try to select the best encapsulation header
- o Evaluate any existing and proposed encapsulations

While the origin and focus of this document is the routing area and in particular NVO3, SFC, and BIER, the considerations apply to other encapsulations that are being defined in the IETF and elsewhere. There seems to be an increase in the number of encapsulations being defined to run over UDP, where there might already exist an encapsulation over IP or Ethernet. Feedback on how these considerations apply in those contexts is welcome.

5. Assumptions

The design center for the new encapsulations is a well-managed network. That network can be a datacenter network (plus datacenter interconnect) or a service provider network. Based on the existing and proposed encapsulations in those environment it is reasonable to make these assumptions:
- o The MTU is carefully managed and configured. Hence an encapsulation protocol can make the packets bigger without resulting in a requirement for fragmentation and reassembly between ingress and egress. (However, it might be useful to detecting MTU misconfigurations.)
- o In general an encapsulation needs some approach for congestion management. But the assumptions are different than for arbitrary Internet paths in that the underlay might be well-provisioned and better policed at the edge, and due to multi-tenancy, the congestion control in the endpoints might be even less trusted than on the Internet at large.

The goal is to implement these encapsulations in hardware and software hence we can’t assume that the needs of either implementation approach can trump the needs of the other. In particular, around extensibility the needs and constraints might be quite different.

6. Terminology

The capitalized keyword MUST is used as defined in http://en.wikipedia.org/wiki/Julmust

TBD: Refer to existing documents for at least NVO3 and SFC terminology. We use at least the VNI ID in this document.

7. Entropy

In many cases the encapsulation format needs to enable ECMP in unmodified routers. Those routers might use different fields in TCP/UDP packets to do ECMP without a risk of reordering a flow. Note that the same entropy might also be used at layer 2 e.g. for Link Aggregation (LAG).

The common way to do ECMP-enabled encapsulation over IP today is to add a UDP header and to use UDP with the UDP source port carrying entropy from the inner/original packet headers as in LISP [RFC6830]. The total entropy consists of 14 bits in the UDP source port (using the ephemeral port range) plus the outer IP addresses which seems to be sufficient for entropy; using outer IPv6 headers would give the option for more entropy should it be needed in the future.

In some environments it might be fine to use all 16 bits of the port range. However, middleboxes might make assumptions about the system ports or user ports. But they should not make any assumptions about the ports in the Dynamic and/or Private Port range, which have the two MSBs set to 11b.

The UDP source port might change over the lifetime of an encapsulated flow, for instance for DoS mitigation or re-balancing load across ECMP. Such changes need to consider reordering if there are packets in flight for the flow.

There is some interaction between entropy and OAM and extensibility mechanism. It is desirable to be able to send OAM packets to follow the same path as network packets. Hence OAM packets should use the same entropy mechanism as data packets. While routers might use information in addition the entropy field and outer IP header, they can not use arbitrary parts of the encapsulation header since that might result in OAM frames taking a different path. Likewise if routers look past the encapsulation header they need to be aware of the extensibility mechanism(s) in the encapsulation format to be able to find the inner headers in the presence of extensions; OAM frames might use some extensions e.g. for timestamps.

Architecturally the entropy and the next header field are really part of enclosing delivery header. UDP with entropy goes hand-in-hand with the outer IP header. Thus the UDP entropy is present for the underlay IP routers the same way that an MPLS entropy label is present for LSRs. The entropy above is all about providing entropy for the outer delivery of the encapsulated packets.

It has been suggested that when IPv6 is used it would not be necessary to add a UDP header for entropy, since the IPv6 flow label can be used for entropy. (This assumes that there is an IP protocol number for the encapsulation in addition to a UDP destination port number since UDP would be used with IPv4 underlay. And any use of UDP checksums would need to be replaced by an encaps-specific checksum or secure hash.) While such an approach would save 8 bytes of headers when the underlay is IPv6, it does assume that the underlay routers use the flow label for ECMP, and it also would make the IPv6 approach different than the IPv4 approach. Currently the leaning is towards recommending using the UDP encapsulation for both IPv4 and IPv6 underlay. The IPv6 flow label can be used for additional entropy if need be. There is more detailed discussion for using the IPv6 flow label for tunnels in [RFC6438].

Note that in the proposed BIER encapsulation [I-D.wijnands-mpls-bier-encapsulation], there is an an 8-bit field which specifies an entropy value that can be used for load balancing purposes. This entropy is for the BIER forwarding decisions, which is independent of any outer delivery ECMP between BIER routers. Thus it is not part of the delivery ECMP discussed in this section.

[Note: For any given bit in BIER (that identifies an exit from the BIER domain) there might be multiple immediate next hops. The BIER entropy field is used to select that next hop as part of BIER processing. The BIER forwarding process may do equal cost load balancing, but the load balancing procedure MUST choose the same path for any two packets that have the same entropy value.]

In summary:

- o The entropy is associated with the transport, that is an outer IP header or MPLS.
- o In the case of IP transport use 14 or 16 bits of UDP source port, plus outer IPv6 flowid for entropy.

8. Next-protocol indication

Next-protocol indications appear in three different contexts for encapsulations.

Firstly, the transport delivery mechanism for the encapsulations we

discuss in this document need some way to indicate which encapsulation header (or other payload) comes next in the packet. Some encapsulations might be identified by a UDP port; others might be identified by an Ethernet type or IP protocol number. Which approach is used is a function of the preceding header the same way as IPv4 is identified by both an Ethernet type and an IP protocol number (for IP-in-IP). In some cases the header type is implicit in some session (L2TP) or path (MPLS) setup. But this is largely beyond the control of the encapsulation protocol. For instance, if there is a requirement to carry the encapsulation after an Ethernet header, then an Ethernet type is needed. If required to be carried after an IP/UDP header, then a UDP port number is needed. For UDP port numbers there are considerations for port number conservation described in [I-D.ietf-tsvwg-port-use].

It is worth mentioning that in the MPLS case of no implicit protocol type many forwarding devices peek at the first nibble of the payload to determine whether to apply IPv4 or IPv6 L3/L4 hashes for load balancing [RFC7325]. That behavior places some constraints on other payloads carried over MPLS and some protocol define an initial control word in the payload with a value of zero in its first nibble [RFC4385] to avoid confusion with IPv4 and IPv6 payload headers.

Secondly, the encapsulation needs to indicate the type of its payload, which is in scope for the design of the encapsulation. We have existing protocols which use Ethernet types (such as GRE). Here each encapsulation header can potentially makes its own choices between:

- o Use the Ethernet type space - makes it easy to carry existing L2 and L3 protocols including IPv4, IPv6, and Ethernet. Disadvantages are that it is a 16 bit number and we probably need far less than 100 values, and the number space is controlled by the IEEE 802 RAC with its own allocation policies.
- o Use the IP protocol number space - makes it easy to carry e.g., ESP in addition to IP and Ethernet but brings in all existing protocol numbers many of which would never be used directly on top of the encapsulation protocol. IANA managed eight bit values, presumably more difficult to get an assigned number than to get a transport port assignment.
- o Define their own next-protocol number space, which can use fewer bits than an Ethernet type and give more flexibility, but at the cost of administering that numbering space (presumably by the IANA).

Thirdly, if the IETF ends up defining multiple encapsulations at about the same time, and there is some chance that multiple such encapsulations can be combined in the same packet, there is a question whether it makes sense to use a common approach and

numbering space for the encapsulation across the different protocols. A common approach might not be beneficial as long as there is only one way to indicate e.g., SFC inside NVO3.

Many Internet protocols use fixed values (typically managed by the IANA function) for their next-protocol field. That facilitates interpretation of packets by middleboxes and e.g., for debugging purposes, but might make the protocol evolution inflexible. Our collective experience with MPLS shows an alternative where the label can be viewed as an index to a table containing processing instructions and the table content can be managed in different ways. Encapsulations might want to consider the tradeoffs between such more flexible versus more fixed approaches.

In summary:

- o Would it be useful for the IETF come up with a common scheme for encapsulation protocols? If not each encapsulation can define its own scheme.

9. MTU and Fragmentation

A common approach today is to assume that the underlay have sufficient MTU to carry the encapsulated packets without any fragmentation and reassembly at the tunnel endpoints. That is sufficient when the operator of the ingress and egress have full control of the paths between those endpoints. And it makes for simpler (hardware) implementations if fragmentation and reassembly can be avoided.

However, even under that assumption it would be beneficial to be able to detect when there is some misconfiguration causing packets to be dropped due to MTU issues. One way to do this is to have the encapsulator set the don’t-fragment (DF) flag in the outer IPv4 header and receive and log any received ICMP "packet too big" (PTB) errors. Note that no flag needs to be set in an outer IPv6 header [RFC2460].

Encapsulations could also define an optional tunnel fragmentation and reassembly mechanism which would be useful in the case when the operator doesn’t have full control of the path, or when the protocol gets deployed outside of its original intended context. Such a mechanism would be required if the underlay might have a path MTU which makes it impossible to carry at least 1518 bytes (if offering Ethernet service), or at least 1280 (if offering IPv6 service). The use of such a protocol mechanism could be triggered by receiving a PTB. But such a mechanism might not be implemented by all encapsulators and decapsulators. [Aerolink is one example of such a

protocol.]

Depending on the payload carried by the encapsulation there are some additional possibilities:

- o If payload is IPv4/6 then the underlay path MTU could be used to report end-to-end path MTU.
- o If the payload service is Ethernet/L2, then there is no such per destination reporting mechanism. However, there is a LLDP TLV for reporting max frame size; might be useful to report minimum to end stations, but unmodified end stations would do nothing with that TLV since they assume that the MTU is at least 1518.

In summary:

- o In some deployments an encapsulation can assume well-managed MTU hence no need for fragmentation and reassembly related to the encapsulation.
- o Even so, it makes sense for ingress to track any ICMP packet too big addressed to ingress to be able to log any MTU misconfigurations.
- o Should an encapsulation protocol be deployed outside of the original context it might very well need support for fragmentation and reassembly.

10. OAM

The OAM area is seeing active development in the IETF with discussions (at least) in NVO3 and SFC working groups, plus the new LIME WG looking at architecture and YANG models.

The design team has take a narrow view of OAM to explore the potential OAM implications on the encapsulation format.

In terms of what we have heard from the various working groups there seem to be needs to:

- o Be able to send out-of-band OAM messages - that potentially should follow the same path through the network as some flow of data packets.
  - * Such OAM messages should not accidentally be decapsulated and forwarded to the end stations.
- o Be able to add OAM information to data packets that are encapsulated. Discussions have been around:
  - * Using a bit in the OAM to synchronize sampling of counters between the encapsulator and decapsulator.
  - * Optional timestamps, sequence numbers, etc for more detailed measurements between encapsulator and decapsulator.

- o Usable for both proactive monitoring (akin to BFD) and reactive checks (akin to traceroute to pin-point a failure)

To ensure that the OAM messages can follow the same path the OAM messages need to get the same ECMP (and LAG hashing) results as a given data flow. An encapsulator can choose between one of:

- o Limit ECMP hashing to not look past the UDP header i.e. the entropy needs to be in the source/destination IP and UDP ports
- o Make OAM packets look the same as data packets i.e. the initial part of the OAM payload has the inner Ethernet, IP, TCP/UDP headers as a payload. (This approach was taken in TRILL out of necessity since there is no UDP header.) Any OAM bit in the encapsulation header must in any case be excluded from the entropy.

There can be several ways to prevent OAM packets from accidentally being forwarded to the end station using:

- o A bit in the frame (as in TRILL) indicating OAM
- o A next-protocol indication with a designated value for "none" or "oam".

This assumes that the bit or next protocol, respectively, would not affect entropy/ECMP in the underlay. However, the next-protocol field might be used to provide differentiated treatment of packets based on their payload; for instance a TCP vs. IPsec ESP payload might be handled differently. Based on that observation it might be undesirable to overload the next protocol with the OAM drop behavior, resulting in a preference for having a bit to indicate that the packet should be forwarded to the end station after decapsulation.

There has been suggestions that one (or more) marker bits in the encaps header would be useful in order to delineate measurement epochs on the encapsulator and decapsulator and use that to compare counters to determine packet loss.

A result of the above is that OAM is likely to evolve and needs some degree of extensibility from the encapsulation format; a bit or two plus the ability to define additional larger extensions.

An open question is how to handle error messages or other reports relating to OAM. One can think if such reporting as being associated with the encapsulation the same way ICMP is associated with IP. Would it make sense for the IETF to develop a common Encapsulation Error Reporting Protocol as part of OAM, which can be used for different encapsulations? And if so, what are the technical challenges. For instance, how to avoid it being filtered as ICMP often is?

A potential additional consideration for OAM is the possible future

existence of gateways that "stitch" together different dataplane encapsulations and might want to carry OAM end-to-end across the different encapsulations.

In summary:
- o It makes sense to reserve a bit for "drop after decapsulation" for OAM out-of-band.
- o An encapsulation needs sufficient extensibility for OAM (such as bits, timestamps, sequence numbers). That might be motivated by in-band OAM but it would make sense to leverage the same extensions for out-of band OAM.
- o OAM places some constraints on use of entropy in forwarding devices.
- o Should IETF look into error reporting that is independent of the specific encapsulation?

11. Security Considerations

Different encapsulation use cases will have different requirements around security. For instance, when encapsulation is used to build overlay networks for network virtualization, isolation between virtual networks may be paramount. BIER support of multicast may entail different security requirements than encapsulation for unicast.

In real deployment, the security of the underlying network may be considered for determining the level of security needed in the encapsulation layer. However for the purposes of this discussion, we assume that network security is out of scope and that the underlying network does not itself provide adequate or as least uniform security mechanisms for encapsulation.

There are at least three considerations for security:
- o Anti-spoofing/virtual network isolation
- o Interaction with packet level security such as IPsec or DTLS
- o Privacy (e.g., VNI ID confidentially for NVO3)

This section uses a VNI ID in NVO3 as an example. A SFC or BIER encapsulation is likely to have fields with similar security and privacy requirements.

11.1. Encapsulation-specific considerations

Some of these considerations appear for a new encapsulation, and others are more specific to network virtualization in datacenters.

- o New attack vectors:
  - * DDOS on specific queued/paths by attempting to reproduce the 5-tuple hash for targeted connections.
  - * Entropy in outer 5-tuple may be too little or predictable.
  - * Leakage of identifying information in the encapsulation header for an encrypted payload.
  - * Vulnerabilities of using global values in fields like VNI ID.
- o Trusted versus untrusted tenants in network virtualization:
  - * The criticality of virtual network isolation depends on whether tenants are trusted or untrusted. In the most extreme cases, tenants might not only be untrusted but may be considered hostile.
  - * For a trusted set of users (e.g. a private cloud) it may be sufficient to have just a virtual network identifier to provide isolation. Packets inadvertently crossing virtual networks should be dropped similar to a TCP packet with a corrupted port being received on the wrong connection.
  - * In the presence of untrusted users (e.g. a public cloud) the virtual network identifier must be adequately protected against corruption and verified for integrity. This case may warrant keyed integrity.
- o Different forms of isolation:
  - * Isolation could be blocking all traffic between tenants (or except as allowed by some firewall)
  - * Could also be about performance isolation i.e. one tenant can overload the network in a way that affects other tenants
  - * Physical isolation of traffic for different tenants in network may be required, as well as required restrictions that tenants may have on where their packets may be routed.
- o New attack vectors from untrusted tenants:
  - * Third party VMs with untrusted tenants allows internally borne attacks within data centers
  - * Hostile VMs inside the system may exist (e.g. public cloud)
  - * Internally launched DDOS
  - * Passive snooping for mis-delivered packets
  - * Mitigate damage and detection in event that a VM is able to circumvent isolation mechanisms
- o Tenant-provider relationship:
  - * Tenant might not trust provider, hypervisors, network
  - * Provider likely will need to provide SLA or a least a statement on security
  - * Tenant may implement their own additional layers of security
  - * Regulation and certification considerations
- o Trend towards tighter security:
  - * Tenants’ data in network increases in volume and value, attacks become more sophisticated

* Large DCs already encrypt everything on disk
* DCs likely to encrypt inter-DC traffic at this point, use TLS to Internet.
* Encryption within DC is becoming more commonplace, becomes ubiquitous when cost is low enough.
* Cost/performance considerations. Cost of support for strong security has made strong network security in DCs prohibitive.
* Are there lessons from MacSec?

11.2. Virtual network isolation

The first requirement is isolation between virtual networks. Packets sent in one virtual network should never be illegitimately received by a node in another virtual network. Isolation should be protected in the presence of malicious attacks or inadvertent packet corruption.

The second requirement is sender authentication. Sender identity is authenticated to prevent anti-spoofing. Even if an attacker has access to the packets in the network, they cannot send packets into a virtual network. This may have two possibilities:

o Pairwise sender authentication. Any two communicating hosts negotiate a shared key.
o Group authentication. A group of hosts share a key (this may be more appropriate for multicast of encapsulation).

Possible security solutions:

o Security cookie: This is similar to L2TP cookie mechanism [RFC3931]. A shared plain text cookie is shared between encapsulator and decapsulator. A receiver validates a packet by evaluating if the cookie is correct for the virtual network and address of a sender. Validation function is F(cookie, VNI ID, source address). If cookie matches, accept packet, else drop. Since cookie is plain text this method does not protect against an eavesdropping. Cookies are set and may be rotated out of band.
o Secure hash: This is a stronger mechanism than simple cookies that borrows from IPsec and PPP authentication methods. In this model security field contains a secure hash of some fields in the packet using a shared key. Hash function may be something like H(key, VNI ID, address, salt). The salt ensures the hash is not the same for every packet, and if it includes a sequence number may also protect against replay attacks.

In any use of a shared key, periodic re-keying should be allowed. This could include use of techniques like generation numbers, key windows, etc. See [I-D.farrelll-mpls-opportunistic-encrypt] for an example application.

We might see firewalls that are aware of the encapsulation and can provide some defense in depth combined with the above example anti-spoofing approaches. An example would be an NVO3-aware firewall being able to check the VNI ID.

Separately and in addition to such filtering, there might be a desire to completely block an encapsulation protocol at certain places in the network, e.g., at the edge of a datacenter. Using a fixed standard UDP destination port number for each encapsulation protocol would facilitate such blocking.

11.3. Packet level security

An encapsulated packet may itself be encapsulated in IPsec (e.g. ESP). This should be straightforward and in fact is what would happen today in security gateways. In this case, there is no special consideration for the fact that packet is encapsulated, however since the encapsulation layer headers are included (part of encrypted data for instance) we lose visibility in the network of the encapsulation.

The more interesting case is when security is applied to the encapsulation payload. This will keep the encapsulation headers in the outer header visible to the network (for instance in nvo3 we may way to firewall based on VNI ID even if the payload is encrypted). One possibility is to apply DTLS to the encapsulation payload. In this model the protocol stack may be something like IP|UDP|Encap|DTLS|encrypted_payload. The encapsulation and security should be done together at an encapsulator and resolved at the decapsulator. Since the encapsulation header is outside of the security coverage, this may itself require security (like described above).

In both of the above the security associations (SAs) may be between physical hosts, so for instance in nvo3 we can have packets of different virtual networks using the same SA-- this should not be an issue since it is the VNI ID that ensures isolation (which needs to be secured also).

11.4. In summary:

- o Encapsulations need extensibility mechanisms to be able to add security features like cookies and secure hashes protecting the encapsulation header.
- o NVO3 probably has specific higher requirements relating to isolation for network virtualization, which is in scope for the NVO3 WG.
- o Our collective IETF experience is that successful protocols get deployed outside of the original intended context, hence the initial assumptions about the threat model might become invalid.

That needs to be considered in the standardization of new encapsulations.

12. QoS

In the Internet architecture we support QoS using the Differentiated Services Code Points (DSCP) in the formerly named Type-of-Service field in the IPv4 header, and in the Traffic-Class field in the IPv6 header. The ToS and TC fields also contain the two ECN bits, which are discussed in Section 13.

We have existing specifications how to process those bits. See [RFC2983] for diffserv handling, which specifies how the received DSCP value is used to set the DSCP value in an outer IP header when encapsulating. (There are also existing specifications how DSCP can be mapped to layer2 priorities.)

Those specifications apply whether or not there is some intervening headers (e.g., for NVO3 or SFC) between the inner and outer IP headers. Thus the encapsulation considerations in this area are mainly about applying the framework in [RFC2983].

Note that the DSCP and ECN bits are not the only part of an inner packet that might potentially affect the outer packet. For example, [RFC2473] specifies handling of inner IPv6 hop-by-hop options that effectively result in copying some options to the outer header. It is simpler to not have future encapsulations depend on such copying behavior.

There are some other considerations specific to doing OAM for encapsulations. If OAM messages are used to measure latency, it would make sense to treat them the same as data payloads. Thus they need to have the same outer DSCP value as the data packets which they wish to measure.

Due to OAM there are constraints on middleboxes in general. If middleboxes inspect the packet past the outer IP+UDP and encapsulation header and look for inner IP and TCP/UDP headers, that might violate the assumption that OAM packets will be handled the same as regular data packets. That issue is broader than just QoS - applies to firewall filters etc.

In summary:

- o Leverage the existing approach in [RFC2983] for DSCP handling.

13. Congestion Considerations

Additional encapsulation headers does not introduce anything new for Explicit Congestion Notification. It is just like IP-in-IP and IPsec tunnels which is specified in [RFC6040] in terms of how the ECN bits in the inner and outer header are handled when encapsulating and decapsulating packets. Thus new encapsulations can more or less include that by reference.

There are additional considerations around carrying non-congestion controlled traffic. These details have been worked out in [I-D.ietf-mpls-in-udp]. As specified in [RFC5405]: "IP-based traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. Consequently, a tunnel carrying IP-based traffic should already interact appropriately with other traffic sharing the path, and specific congestion control mechanisms for the tunnel are not necessary". Those considerations are being captured in [I-D.ietf-tsvwg-rfc5405bis].

For this reason, where an encapsulation method is used to carry IP traffic that is known to be congestion controlled, the UDP tunnels does not create an additional need for congestion control. Internet IP traffic is generally assumed to be congestion-controlled. Similarly, in general Layer 3 VPNs are carrying IP traffic that is similarly assumed to be congestion controlled.

However, some of the encapsulations (at least NVO3) will be able to carry arbitrary Layer 2 packets to provide an L2 service, in which case one can not assume that the traffic is congestion controlled.

One could handle this by adding some congestion control support to the encapsulation header (one instance of which would end up looking like DCCP). However, if the underlay is well-provisioned and managed as opposed to being arbitrary Internet path, it might be sufficient to have a slower reaction to congestion induced by that traffic. There is work underway on a notion of "circuit breakers" for this purpose. See See [I-D.ietf-tsvwg-circuit-breaker]. Encapsulations which carry arbitrary Layer 2 packets want to consider that ongoing work.

If the underlay is provisioned in such a way that it can guarantee sufficient capacity for non-congestion controlled Layer 2 traffic, then such circuit breakers might not be needed.

Two other considerations appear in the context of these encapsulations as applied to overlay networks:

- o Protect against malicious end stations
- o Ensure fairness and/or measure resource usage across multiple tenants

Those issues are really orthogonal to the encapsulation, in that they are present even when no new encapsulation header is in use. However, the application of the new encapsulations are likely to be in environments where those issues are becoming more important. Hence it makes sense to consider them.

One could make the encapsulation header be extensible to that it can carry sufficient information to be able to measure resource usage, delays, and congestion. The suggestions in the OAM section about a single bit for counter synchronization, and optional timestamps and/or sequence numbers, could be part of such an approach. There might also be additional congestion-control extensions to be carried in the encapsulation. Overall this results in a consideration to support sufficient extensibility in the encapsulation to handle potential future developments in this space.

Coarse measurements are likely to suffice, at least for circuit-breaker-like purposes, see [I-D.wei-tsvwg-tunnel-congestion-feedback] and [I-D.briscoe-conex-data-centre] for examples on active work in this area via use of ECN. [RFC6040] Appendix C is also relevant. The outer ECN bits seem sufficient (at least when everything uses ECN) to do this course measurements. Needs some more study for the case when there are also drops; might need to exchange counters between ingress and egress to handle drops.

Circuit breakers are not sufficient to make a network with different congestion control when the goal is to provide a predictable service to different tenants. The fallback would be to rate limit different traffic.

In summary:

- o Leverage the existing approach in [RFC6040] for ECN handling.
- o If the encapsulation can carry non-IP, hence non-congestion controlled traffic, then leverage the approach in [I-D.ietf-mpls-in-udp].
- o "Watch this space" for circuit breakers.

14. Header Protection

Many UDP based encapsulations such as VXLAN [RFC7348] either discourage or explicitly disallow the use of UDP checksums. The reason is that the UDP checksum covers the entire payload of the packet and switching ASICs are typically optimized to look at only a small set of headers as the packet passes through the switch. In

these case, computing a checksum over the packet is very expensive. (Software endpoints and the NICs used with them generally do not have the same issue as they need to look at the entire packet anyways.)

The lack a header checksum creates the possibility that bit errors can be introduced into any information carried by the new headers. Specifically, in the case of IPv6, the assumption is that a transport layer checksum - UDP in this case - will protect the IP addresses through the inclusion of a pseudo-header in the calculation. This is different from IPv4 on which many of these encapsulation protocols are initially deployed which contains its own header checksum. In addition to IP addresses, the encapsulation header often contains its own information which is used for addressing packets or other high value network functions. Without a checksum, this information is potentially vulnerable - an issue regardless of whether the packet is carried over IPv4 or IPv6.

Several protocols cite [RFC6935] and [RFC6936] as an exemption to the IPv6 checksum requirements. However, these are intended to be tailored to a fairly narrow set of circumstances - primarily relying on sparseness of the address space to detect invalid values and well managed networks - and are not a one size fits all solution. In these cases, an analysis should be performed of the intended environment, including the probability of errors being introduced and the use of ECC memory in routing equipment.

Conceptually, the ideal solution to this problem is a checksum that covers only the newly added headers of interest. There is little value in the portion of the UDP checksum that covers the encapsulated packet because that would generally be protected by other checksums and this is the expensive portion to compute. In fact, this solution already exists in the form of UDP-Lite and UDP based encapsulations could be easily ported to run on top of it. Unfortunately, the main value in using UDP as part of the encapsulation header is that it is recognized by already deployed equipment for the purposes of ECMP, RSS, and middlebox operations. As UDP-Lite uses a different protocol number than UDP and it is not widely implemented in middleboxes, this value is lost. A possible solution is to incorporate the same partial-checksum concept as UDP-Lite or other header checksum protection into the encapsulation header and continue using UDP as the outer protocol. One potential challenge with this approach is the use of NAT or other form of translation on the outer header will result in an invalid checksum as the translator will not know to update the encapsulation header.

The method chosen to protect headers is often related to the security needs of the encapsulation mechanism. On one hand, the impact of a poorly protected header is not limited to only data corruption but

can also introduce a security vulnerability in the form of misdirected packets to an unauthorized recipient. Conversely, high security protocols that already include a secure hash over the valuable portion of the header (such as by encrypting the entire IP packet using IPsec, or some secure hash of the encap header) do not require additional checksum protection as the hash provides stronger assurance than a simple checksum.

If the sender has included a checksum, then the receiver should verify that checksum or, if incapable, drop the packet. The assumption is that configuration and/or control-plane capability exchanges can be used when different receiver have different checksum validation capabilities.

In summary:
- o Encapsulations need extensibility to be able to add checksum/CRC for the encapsulation header itself.
- o When the encapsulation has a checksum/CRC, include the IPv6 pseudo-header in it.
- o The checksum/CRC can potentially be avoided when cryptographic protection is applied to the encapsulation.

15. Extensibility Considerations

Protocol extensibility is the concept that a networking protocol may be extended to include new use cases or functionality that were not part of the original protocol specification. Extensibility may be used to add security, control, management, or performance features to a protocol. A solution may allow private extensions for customization or experimentation.

Extending a protocol often implies that a protocol header must carry new information. There are two usual methods to accomplish this:
1. Define or redefine the meaning of existing fields in a protocol header.
2. Add new (optional) fields to the protocol header.

It is also possible to create a new protocol version, but this is more associated with defining a protocol than extending it (IPv6 being a successor to IPv4 is an example of protocol versioning).

In some cases it might be more appropriate to define a new inner protocol which can carry the new functionality instead of extending the outer protocol. Examples where this works well is in the IP/transport split, where the earlier architecture had a single NCP [RFC0033] protocol which carried both the hop-by-hop semantics which are now in IP, and the end-to-end semantics which are now in TCP. Such a split is effective when different nodes need to act upon the

different information. Applying this for general protocol extensibility through nesting is not well understood, and does result in longer header chains. Furthermore, our experience with IPv6 extension headers [RFC2460] in middleboxes indicates that the header chaining approach does not help with middlebox traversal.

Many protocol definitions include some number of reserved fields or bits which can be used for future extension. VXLAN is an example of a protocol that includes reserved bits which are subsequently being allocated for new purposes. Another technique employed is to re-purpose existing header fields with new meanings. A classic example of this is the definition of DSCP code point which redefines the ToS field originally specified in IPv4. When a field is redefined, some mechanism may be needed to ensure that all interested parties agree on the meaning of the field. The techniques of defining meaning for reserved bits or redefining existing fields have the advantage that a protocol header can be kept a fixed length. The disadvantage is that the extensibility is limited. For instance, the number reserved bits in a fixed protocol header is limited. For standard protocols the decision to commit to a definition for a field can be wrenching since it is difficult to retract later. Also, it is difficult to predict a priori how many reserved fields or bits to put into a protocol header to satisfy the extensions create over the lifetime of the protocol.

Extending a protocol header with new fields can be done in several ways.

- o TLVs are a very popular method used in such protocols as IP and TCP. Depending on the type field size and structure, TLVs can offer a virtually unlimited range of extensions. A disadvantage of TLVs is that processing them can be verbose, quite complicated, several validations must often be done for each TLV, and there is no deterministic ordering for a list of TLVs. TCP serves as an example of a protocol where TLVs have been successfully used (i.e. required for protocol operation). IP is an example of a protocol that allows TLVs but are rarely used in practice (router fast paths usually that assume no IP options). Note that TCP TLVs are implemented in software as well as (NIC) hardware handling various forms of TCP offload. Additional discussions about hardware implications for extensibility is captured in Section 18.
- o Extension headers are closely related to TLVs. These also carry type/value information, but instead of being a list of TLVs within a single protocol header, each one is in its own protocol header. IPv6 extension headers and SFC NSH are examples of this technique. Similar to TLVs these offer a wide range of extensibility, but have similarly complex processing. Another difference with TLVs is that each extension header is idempotent. This is beneficial in cases where a protocol implements a push/pop model for header elements like service chaining, but makes it more difficult group

correlated information within one protocol header.

- o A particular form of extension headers are the tags used by IEEE 802 protocols. Those are similar to e.g., IPv6 extension headers but with the key difference that each tag is a fixed length header where the length is implicit in the tag value. Thus as long as a receiver can be programmed with a tag value to length map, it can skip those new tags.
- o Flag-fields are a non-TLV like method of extending a protocol header. The basic idea is that the header contains a set of flags, where each set flags corresponds to optional field that is present in the header. GRE is an example of a protocol that employs this mechanism. The fields are present in the header in the order of the flags, and the length of each field is fixed. Flag-fields are simpler to process compared to TLVs, having fewer validations and the order of the optional fields is deterministic. A disadvantage is that range of possible extensions with flag-fields is smaller than TLVs.

The requirements for receiving unknown or unimplemented extensible elements in an encapsulation protocol (flags, TLVs, optional fields) need to be specified. There are two parties to consider, middle boxes and terminal endpoints of encapsulation (at the decapsulator).

A protocol may allow or expect nodes in a path to modify fields in an encapsulation (example use of this is BIER). In this case, the middleboxes should follow the same requirements as nodes terminating the encapsulation. In the case that middle boxes do not modify the encapsulation, we can assume that they may still inspect any fields of the encapsulation. Missing or unknown fields should be accepted per protocol specification, however it is permissible for a site to implement a local policy otherwise (e.g. a firewall may drop packets with unknown options).

For handling unknown options at terminal nodes, there are two possibilities: drop packet or accept while ignoring the unknown options. Many Internet protocols specify that reserved flags must be set to zero on transmission and ignored on reception. L2TP is example data protocol that has such flags. GRE is a notable exception to this rule, reserved flag bits 1-5 cannot be ignored [RFC2890]. For TCP and IPv4, implementations must ignore optional TLVs with unknown type; however in IPv6 if a packet contains an unknown extension header (unrecognized next header type) the packet must be dropped with an ICMP error message returned. The IPv6 options themselves (encoded inside the destinations options or hop-by-hop options extension header) have more flexibility. There are bits in the option code are used to instruct the receiver whether to ignore, silently drop, or drop and send error if the option is unknown. Some protocols define a "mandatory bit" that can is set

with TLVs to indicate that an option must not be ignored. Conceptually, optional data elements can only be ignored if they are idempotent and do not alter how the rest of the packet is parsed or processed.

Depending on what type of protocol evolution one can predict, it might make sense to have a way for a sender to express that the packet should be dropped by a terminal node which does not understand the new information. In other cases it would make sense to have the receiver silently ignore the new info. The former can be expressed by having a version field in the encapsulation, or a notion of "mandatory bit" as discussed above.

A security mechanism which use some form secure hash over the encapsulation header would need to be able to know which extensions can be changed in flight.

In summary:

- o Encapsulations need the ability to be extended to handle e.g., the OAM or security aspects discussed in this document.
- o Practical experience seems to tell us that extensibility mechanisms which are not in use on day one might result in immediate ossification by lack of implementation support. In some cases that has occurred in routers and in other cases in middleboxes. Hence devising ways where the extensibility mechanisms are in use seems important.

16. Layering Considerations

One can envision that SFC might use NVO3 as a delivery/transport mechanism. With more imagination that in turn might be delivered using BIER. Thus it is useful to think about what things look like when we have BIER+NVO3+SFC+payload. Also, if NVO3 is widely deployed there might be cases of NVO3 nesting where a customer uses NVO3 to provide network virtualization e.g., across departments. That customer uses a service provider which happens to use NVO3 to provide transport for their customers.Thus NVO3 in NVO3 might happen.

A key question we set out to answer is what the packets might look like in such a case, and in particular whether we would end up with multiple UDP headers for entropy.

Based on the discussion in the Entropy section, the entropy is associated with the outer delivery IP header. Thus if there are multiple IP headers there would be a UDP header for each one of the IP headers. But SFC does not require its own IP header. So a case of NVO3+SFC would be IP+UDP+NVO3+SFC. A nested NVO3 encapsulation

would have independent IP+UDP headers.

The layering also has some implications for middleboxes.

- o A device on the path between the ingress and egress is allowed to transparently inspect all layers of the protocol stack and drop or forward, but not transparently modify anything but the layer in which they operate. What this means is that an IP router is allowed modify the outer IP ttl and ECN bits, but not the encapsulation header or inner headers and payload. And a BIER router is allowed to modify the BIER header.
- o Alternatively such a device can become visible at a higher layer. E.g., a middlebox could a middlebox could first decapsulate, perform some function then encapsulate; which means it will generate a new encapsulation header.

The design team asked itself some additional questions:

- o Would it make sense to have a common encapsulation base header (for OAM, security?, etc) and then followed by the specific information for NVO3, SFC, BIER? Given that there are separate proposals and the set of information needing to be carried differs, and the extensibility needs might be different, it would be difficult and not that useful to have a common base header.
- o With a base header in place, one could view the different functions (NVO3, SFC, and BIER) as different extensions to that base header resulting in encodings which are more space optimal by not repeating the same base header. The base header would only be repeated when there is an additional IP (and hence UDP) header. That could mean a single length field (to skip to get to the payload after all the encapsulation headers). That might be technically feasible, but it would create a lot of dependencies between different WGs making it harder to make progress. Compare with the potential savings in packet size.

17. Service model

The IP service is lossy and subject to reordering. In order to avoid a performance impact on transports like TCP the handling of packets is designed to avoid reordering packets that are in the same transport flow (which is typically identified by the 5-tuple). But across such flows the receiver can see different ordering for a given sender. That is the case for a unicast vs. a multicast flow from the same sender.

There is a general tussle between the desire for high capacity utilization across a multipath network and the impact on packet ordering within the same flow (which results in lower transport protocol performance). That isn’t affected by the introduction of an

encapsulation. However, the encapsulation comes with some entropy, and there might be cases where folks want to change that in response to overload or failures. For instance, one might want to change UDP source port to try different ECMP route. Such changes can result in packet reordering within a flow, hence would need to be done infrequently and with care e.g., by identifying packet trains.

There might be some applications/services which are not able to handle reordering across flows. The IETF has defined pseudo-wires [RFC3985] which provides the ability to ensure ordering (implemented using sequence numbers and/or timestamps).

Architectural such services would make sense, but as a separate layer on top of an encapsulation protocol. They could be deployed between ingress and egress of a tunnel which uses some encaps. Potentially the tunnel control points at the ingress and egress could become a platform for fixing suboptimal behavior elsewhere in the network. That would clearly be undesirable in the general case. However, handling encapsulation of non-IP traffic hence non-congestion-controlled traffic is likely to be required, which implies some fairness and/or QoS policing on the ingress and egress devices.

But the tunnels could potentially do more like increase reliability (retransmissions, FEC) or load spreading using e.g. MP-TCP between ingress and egress.

18. Hardware Friendly

Hosts, switches and routers often leverage capabilities in the hardware to accelerate packet encapsulation, decapsulation and forwarding.

Some design considerations in encapsulation that leverage these hardware capabilities may result in more efficiently packet processing and higher overall protocol throughput.

While "hardware friendliness" can be viewed as unnecessary considerations for a design, part of the motivation for considering this is ease of deployment; being able to leverage existing NIC and switch chips for at least a useful subset of the functionality that the new encapsulation provides. The other part is the ease of implementing new NICs and switch/router chips that support the encapsulation at ever increasing line rates.

[disclaimer] There are many different types of hardware in any given network, each maybe better at some tasks while worse at others. We would still recommend protocol designers to examine the specific

hardware that are likely to be used in their networks and make decisions on a case by case basis.

Some considerations are:

- o Keep the encap header small. Switches and routers usually only read the first small number of bytes into the fast memory for quick processing and easy manipulation. The bulk of the packets are usually stored in slow memory. A big encap header may not fit and additional read from the slow memory will hurt the overall performance and throughput.
- o Put important information at the beginning of the encapsulation header. The reasoning is similar as explained in the previous point. If important information are located at the beginning of the encapsulation header, the packet may be processed with smaller number of bytes to be read into the fast memory and improve performance.
- o Avoid full packet checksums in the encapsulation if possible. Encapsulations should instead consider adding their own checksum which covers the encapsulation header and any IPv6 pseudo-header. The motivation is that most of the switch/router hardware make switching/forwarding decisions by reading and examining only the first certain number of bytes in the packet. Most of the body of the packet do not need to be processed normally. If we are concerned of preventing packet to be misdelivered due to memory errors, consider only perform header checksums. Note that NIC chips can typically already do full packet checksums for TCP/UDP, while adding a header checksum might require adding some hardware support.
- o Place important information at fixed offset in the encapsulation header. Packet processing hardware may be capable of parallel processing. If important information can be found at fixed offset, different part of the encapsulation header may be processed by different hardware units in parallel (for example multiple table lookups may be launched in parallel). It is easier for hardware to handle optional information when the information, if present, can be found in ideally one place, but in general, in as few places as possible. That facilitates parallel processing. TLV encoding with unconstrained order typically does not have that property.
- o Limit the number of header combinations. In many cases the hardware can explore different combinations of headers in parallel, however there is some added cost for this.

18.1. Considerations for NIC offload

This section provides guidelines to provide support of common offloads for encapsulation in Network Interface Cards (NICs). Offload mechanisms are techniques that are implemented separately

from the normal protocol implementation of a host networking stack and are intended to optimize or speed up protocol processing. Hardware offload is performed within a NIC device on behalf of a host.

There are three basic offload techniques of interest:
- o Receive multi queue
- o Checksum offload
- o Segmentation offload

18.1.1. Receive multi-queue

Contemporary NICs support multiple receive descriptor queues (multi-queue). Multi-queue enables load balancing of network processing for a NIC across multiple CPUs. On packet reception, a NIC must select the appropriate queue for host processing. Receive Side Scaling (RSS) is a common method which uses the flow hash for a packet to index an indirection table where each entry stores a queue number.

UDP encapsulation, where the source port is used for entropy, should be compatible with multi-queue NICs that support five-tuple hash calculation for UDP/IP packets as input to RSS. The source port ensures classification of the encapsulated flow even in the case that the outer source and destination addresses are the same for all flows (e.g. all flows are going over a single tunnel).

18.1.2. Checksum offload

Many NICs provide capabilities to calculate standard ones complement payload checksum for packets in transmit or receive. When using encapsulation over UDP there are at least two checksums that may be of interest: the encapsulated packet’s transport checksum, and the UDP checksum in the outer header.

18.1.2.1. Transmit checksum offload

NICs may provide a protocol agnostic method to offload transmit checksum (NETIF_F_HW_CSUM in Linux parlance) that can be used with UDP encapsulation. In this method the host provides checksum related parameters in a transmit descriptor for a packet. These parameters include the starting offset of data to checksum, the length of data to checksum, and the offset in the packet where the computed checksum is to be written. The host initializes the checksum field to pseudo header checksum. In the case of encapsulated packet, the checksum for an encapsulated transport layer packet, a TCP packet for instance, can be offloaded by setting the appropriate checksum parameters.

NICs typically can offload only one transmit checksum per packet, so simultaneously offloading both an inner transport packet's checksum and the outer UDP checksum is likely not possible. In this case setting UDP checksum to zero (per above discussion) and offloading the inner transport packet checksum might be acceptable.

There is a proposal in [I-D.herbert-remotecsumoffload] to leverage NIC checksum offload when an encapsulator is co-resident with a host.

18.1.2.2. Receive checksum offload

Protocol encapsulation is compatible with NICs that perform a protocol agnostic receive checksum (CHECKSUM_COMPLETE in Linux parlance). In this technique, a NIC computes a ones complement checksum over all (or some predefined portion) of a packet. The computed value is provided to the host stack in the packet's receive descriptor. The host driver can use this checksum to "patch up" and validate any inner packet transport checksum, as well as the outer UDP checksum if it is non-zero.

Many legacy NICs don't provide checksum-complete but instead provide an indication that a checksum has been verified (CHECKSUM_UNNECESSARY in Linux). Usually, such validation is only done for simple TCP/IP or UDP/IP packets. If a NIC indicates that a UDP checksum is valid, the checksum-complete value for the UDP packet is the "not" of the pseudo header checksum. In this way, checksum-unnecessary can be converted to checksum-complete. So if the NIC provides checksum-unnecessary for the outer UDP header in an encapsulation, checksum conversion can be done so that the checksum-complete value is derived and can be used by the stack to validate an checksums in the encapsulated packet.

18.1.3. Segmentation offload

Segmentation offload refers to techniques that attempt to reduce CPU utilization on hosts by having the transport layers of the stack operate on large packets. In transmit segmentation offload, a transport layer creates large packets greater than MTU size (Maximum Transmission Unit). It is only at much lower point in the stack, or possibly the NIC, that these large packets are broken up into MTU sized packet for transmission on the wire. Similarly, in receive segmentation offload, small packets are coalesced into large, greater than MTU size packets at a point low in the stack receive path or possibly in a device. The effect of segmentation offload is that the number of packets that need to be processed in various layers of the stack is reduced, and hence CPU utilization is reduced.

18.1.3.1. Transmit Segmentation Offload

Transmit Segmentation Offload (TSO) is a NIC feature where a host provides a large (larger than MTU size) TCP packet to the NIC, which in turn splits the packet into separate segments and transmits each one. This is useful to reduce CPU load on the host.

The process of TSO can be generalized as:

- o Split the TCP payload into segments which allow packets with size less than or equal to MTU.
- o For each created segment:
  1. Replicate the TCP header and all preceding headers of the original packet.
  2. Set payload length fields in any headers to reflect the length of the segment.
  3. Set TCP sequence number to correctly reflect the offset of the TCP data in the stream.
  4. Recompute and set any checksums that either cover the payload of the packet or cover header which was changed by setting a payload length.

Following this general process, TSO can be extended to support TCP encapsulation UDP. For each segment the Ethernet, outer IP, UDP header, encapsulation header, inner IP header if tunneling, and TCP headers are replicated. Any packet length header fields need to be set properly (including the length in the outer UDP header), and checksums need to be set correctly (including the outer UDP checksum if being used).

To facilitate TSO with encapsulation it is recommended that optional fields should not contain values that must be updated on a per segment basis-- for example an encapsulation header should not include checksums, lengths, or sequence numbers that refer to the payload. If the encapsulation header does not contain such fields then the TSO engine only needs to copy the bits in the encapsulation header when creating each segment and does not need to parse the encapsulation header.

18.1.3.2. Large Receive Offload

Large Receive Offload (LRO) is a NIC feature where packets of a TCP connection are reassembled, or coalesced, in the NIC and delivered to the host as one large packet. This feature can reduce CPU utilization in the host.

LRO requires significant protocol awareness to be implemented correctly and is difficult to generalize. Packets in the same flow need to be unambiguously identified. In the presence of tunnels or

network virtualization, this may require more than a five-tuple match (for instance packets for flows in two different virtual networks may have identical five-tuples). Additionally, a NIC needs to perform validation over packets that are being coalesced, and needs to fabricate a single meaningful header from all the coalesced packets.

The conservative approach to supporting LRO for encapsulation would be to assign packets to the same flow only if they have identical five-tuple and were encapsulated the same way. That is the outer IP addresses, the outer UDP ports, encapsulated protocol, encapsulation headers, and inner five tuple are all identical.

18.1.3.3. In summary:

In summary, for NIC offload:

- o The considerations for using full UDP checksums are different for NIC offload than for implementations in forwarding devices like routers and switches.
- o Be judicious about encapsulations that change fields on a per-packet basis, since such behavior might make it hard to use TSO.

19. Middlebox Considerations

This document has touched upon middleboxes in different section. The reason for this is as encapsulations get widely deployed one would expect different forms of middleboxes might become aware of the encapsulation protocol just as middleboxes have been made aware of other protocols where there are business and deployment opportunities. Such middleboxes are likely to do more than just drop packets based on the UDP port number used by an encapsulation protocol.

We note that various forms of encapsulation gateways that stitch one encapsulation protocol together with another form of protocol could have similar effects.

An example of a middlebox that could see some use would be an NVO3-aware firewall that would filter on the VNI IDs to provide some defense in depth inside or across NVO3 datacenters.

A question for the IETF is whether we should document what to do or what not to do in such middleboxes. This document touches on areas of OAM and ECMP as it relates to middleboxes and it might make sense to document how encapsulation-aware middleboxes should avoid unintended consequences in those (and perhaps other) areas.

In summary:

- o We are likely to see middleboxes that at least parse the headers for successful new encapsulations.
- o Should the IETF document considerations for what not to do in such middleboxes?

20. Related Work

The IETF and industry has defined encapsulations for a long time, with examples like GRE [RFC2890], VXLAN [RFC7348], and NVGRE [I-D.sridharan-virtualization-nvgre] being able to carry arbitrary Ethernet payloads, and various forms of IP-in-IP and IPsec encapsulations that can carry IP packets. As part of NVO3 there has been additional proposals like Geneve [I-D.gross-geneve] and GUE [I-D.herbert-gue] which look at more extensibility. NSH [I-D.quinn-sfc-nsh] is an example of an encapsulation that tries to provide extensibility mechanisms which target both hardware and software implementations.

There is also a large body of work around MPLS encapsulations [RFC3032]. The MPLS-in-UDP work [I-D.ietf-mpls-in-udp] and GRE over UDP [I-D.ietf-tsvwg-gre-in-udp-encap] have worked on some of the common issues around checksum and congestion control. MPLS also introduced a entropy label [RFC6790]. There is also a proposal for MPLS encryption [I-D.farrelll-mpls-opportunistic-encrypt].

The idea to use a UDP encapsulation with a UDP source port for entropy for the underlay routers' ECMP dates back to LISP [RFC6830].

The pseudo-wire work [RFC3985] is interesting in the notion of layering additional services/characteristics such as ordered delivery or timely deliver on top of an encapsulation. That layering approach might be useful for the new encapsulations as well. For instance, the control word [RFC4385]. There is also material on congestion control for pseudo-wires in [I-D.ietf-pwe3-congcons].

Both MPLS and L2TP [RFC3931] rely on some control or signaling to establish state (for the path/labels in the case of MPLS, and for the session in the case of L2TP). The NVO3, SFC, and BIER encapsulations will also have some separation between the data plane and control plane, but the type of separation appears to be different.

IEEE 802.1 has defined encapsulations for L2 over L2, in the form of Provider backbone Bridging (PBB) [IEEE802.1Q-2014] and Equal Cost Multipath (ECMP) [IEEE802.1Q-2014]. The latter includes something very similar to the way the UDP source port is used as entropy: "The flow hash, carried in an F-TAG, serves to distinguish frames belonging to different flows and can be used in the forwarding

process to distribute frames over equal cost paths"

TRILL, which is also a L2 over L2 encapsulation, took a different approach to entropy but preserved the ability for OAM frames [RFC7174] to use the same entropy hence ECMP path as data frames. In [I-D.ietf-trill-oam-fm] there 96 bytes of headers for entropy in the OAM frames, followed by the actual OAM content. This ensures that any headers, which fit in those 96 bytes except the OAM bit in the TRILL header, can be used for ECMP hashing.

As encapsulations evolve there might be a desire to fit multiple inner packets into one outer packet. The work in [I-D.saldana-tsvwg-simplemux] might be interesting for that purpose.

21. Acknowledgements

The authors acknowledge the comments from Alia Atlas, Fred Baker, David Black, Bob Briscoe, Stewart Bryant, Mike Cox, Andy Malis, Radia Perlman, Carlos Pignataro, Jamal Hadi Salim, Michael Smith, and Lucy Yong.

22. Open Issues

- o Middleboxes:
  - * Due to OAM there are constraints on middleboxes in general. If middleboxes inspect the packet past the outer IP+UDP and encapsulation header and look for inner IP and TCP/UDP headers, that might violate the assumption that OAM packets will be handled the same as regular data packets. That issue is broader than just QoS - applies to firewall filters etc.
  - * Firewalls looking at inner payload? How does that work for OAM frames? Even if it only drops ... TRILL approach might be an option? Would that encourage more middleboxes making the network more fragile?
  - * Editorially perhaps we should pull the above two into a separate section about middlebox considerations?
- o Next-protocol indication - should it be common across different encapsulation headers? We will have different ways to indicate the presence of the first encapsulation header in a packet (could be a UDP destination port, an Ethernet type, etc depending on the outer delivery header). But for the next protocol past an encapsulation header one could envision creating or adoption a common scheme. Such a would also need to be able to identify following headers like Ethernet, IPv4/IPv6, ESP, etc.

- o Common OAM error reporting protocol?
- o There is discussion about timestamps, sequence numbers, etc in three different parts of the document. OAM, Congestion Considerations, and Service Model, where the latter argues that a pseudo-wire service should really be layered on top of the encapsulation using its own header. Those recommendations seem to be at odds with each other. Do we envision sequence numbers, timestamps, etc as potential extensions for OAM and CC? If so, those extensions could be used to provide a service which doesn't reorder packets.

23. Change Log

The changes from draft-rtg-dt-encap-01 based on feedback at the Dallas IETF meeting:

- o Setting the context that not all common issues might apply to all encapsulations, but that they should all be understood before being dismissed.
- o Clarified that IPv6 flow label is useful for entropy in combination with a UDP source port.
- o Editorially added a "summary" set of bullets to most sections.
- o Editorial clarifications in the next protocol section to more clearly state the three areas.
- o Folded the two next protocol sections into one.
- o Mention the MPLS first nibble issue in the next protocol section.
- o Mention that viewing the next protocol as an index to a table with processing instructions can provide additional flexibility in the protocol evolution.
- o For the OAM "don't forward to end stations" added that defining a bit seems better than using a special next-protocol value.
- o Added mention of DTLS in addition to IPsec for security.
- o Added some mention of IPv6 hob-by-hop options of other headers than potentially can be copied from inner to outer header.
- o Added text on architectural considerations when it might make sense to define an additional header/protocol as opposed to using the extensibility mechanism in the existing encapsulation protocol.
- o Clarified the "unconstrained TLVs" in the hardware friendly section.
- o Clarified the text around checksum verification and full vs. header checksums.
- o Added wording that the considerations might apply for encaps outside of the routing area.
- o Added references to draft-ietf-pwe3-congcons, draft-ietf-tsvwg-rfc5405bis, RFC2473, and RFC7325

- o Removed reference to RFC3948.
- o Updated the acknowledgements section.
- o Added this change log section.

24. References

24.1. Normative References

[I-D.ietf-tsvwg-rfc5405bis]
Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", draft-ietf-tsvwg-rfc5405bis-02 (work in progress), April 2015.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, September 2000.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC3931] Lau, J., Townsley, M., and I. Goyret, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, March 2005.

[RFC3985] Bryant, S. and P. Pate, "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, February 2006.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label

for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, November 2011.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, April 2013.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.

[RFC7174] Salam, S., Senevirathne, T., Aldrin, S., and D. Eastlake, "Transparent Interconnection of Lots of Links (TRILL) Operations, Administration, and Maintenance (OAM) Framework", RFC 7174, May 2014.

[RFC7325] Villamizar, C., Kompella, K., Amante, S., Malis, A., and C. Pignataro, "MPLS Forwarding Compliance and Performance Requirements", RFC 7325, August 2014.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.

[RFC7364] Narten, T., Gray, E., Black, D., Fang, L., Kreeger, L., and M. Napierala, "Problem Statement: Overlays for Network Virtualization", RFC 7364, October 2014.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, October 2014.

24.2. Informative References

[I-D.briscoe-conex-data-centre]
Briscoe, B. and M. Sridharan, "Network Performance Isolation in Data Centres using Congestion Policing", draft-briscoe-conex-data-centre-02 (work in progress), February 2014.

[I-D.farrelll-mpls-opportunistic-encrypt]
Farrel, A. and S. Farrell, "Opportunistic Security in MPLS Networks", draft-farrelll-mpls-opportunistic-encrypt-05 (work in progress), June 2015.

[I-D.gross-geneve]
Gross, J., Sridhar, T., Garg, P., Wright, C., Ganga, I., Agarwal, P., Duda, K., Dutt, D., and J. Hudson, "Geneve: Generic Network Virtualization Encapsulation", draft-gross-geneve-02 (work in progress), October 2014.

[I-D.herbert-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-herbert-gue-03 (work in progress), March 2015.

[I-D.herbert-remotecsumoffload]
Herbert, T., "Remote checksum offload for encapsulation", draft-herbert-remotecsumoffload-01 (work in progress), November 2014.

[I-D.ietf-mpls-in-udp]
Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", draft-ietf-mpls-in-udp-11 (work in progress), January 2015.

[I-D.ietf-nvo3-arch]
Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Overlay Networks (NVO3)", draft-ietf-nvo3-arch-03 (work in progress), March 2015.

[I-D.ietf-pwe3-congcons]
Stein, Y., Black, D., and B. Briscoe, "Pseudowire Congestion Considerations", draft-ietf-pwe3-congcons-02 (work in progress), July 2014.

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-09 (work in progress), June 2015.

[I-D.ietf-sfc-problem-statement]
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-13 (work in progress), February 2015.

[I-D.ietf-trill-oam-fm]
Senevirathne, T., Finn, N., Salam, S., Kumar, D.,

Eastlake, D., Aldrin, S., and L. Yizhou, "TRILL Fault Management", draft-ietf-trill-oam-fm-11 (work in progress), October 2014.

[I-D.ietf-tsvwg-circuit-breaker]
Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-01 (work in progress), March 2015.

[I-D.ietf-tsvwg-gre-in-udp-encap]
Crabbe, E., Yong, L., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", draft-ietf-tsvwg-gre-in-udp-encap-07 (work in progress), July 2015.

[I-D.ietf-tsvwg-port-use]
Touch, J., "Recommendations on Using Assigned Transport Port Numbers", draft-ietf-tsvwg-port-use-11 (work in progress), April 2015.

[I-D.quinn-sfc-nsh]
Quinn, P., Guichard, J., Surendra, S., Smith, M., Henderickx, W., Nadeau, T., Agarwal, P., Manur, R., Chauhan, A., Halpern, J., Majee, S., Elzur, U., Melman, D., Garg, P., McConnell, B., Wright, C., and K. Kevin, "Network Service Header", draft-quinn-sfc-nsh-07 (work in progress), February 2015.

[I-D.saldana-tsvwg-simplemux]
Saldana, J., "Simplemux. A generic multiplexing protocol", draft-saldana-tsvwg-simplemux-02 (work in progress), January 2015.

[I-D.shepherd-bier-problem-statement]
Shepherd, G., Dolganow, A., and a. arkadiy.gulko@thomsonreuters.com, "Bit Indexed Explicit Replication (BIER) Problem Statement", draft-shepherd-bier-problem-statement-02 (work in progress), February 2015.

[I-D.sridharan-virtualization-nvgre]
Garg, P. and Y. Wang, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-08 (work in progress), April 2015.

[I-D.wei-tsvwg-tunnel-congestion-feedback]
Wei, X., Zhu, L., Deng, L., and B. Briscoe, "Tunnel Congestion Feedback",

draft-wei-tsvwg-tunnel-congestion-feedback-04 (work in progress), June 2015.

[I-D.wijnands-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-wijnands-bier-architecture-05 (work in progress), March 2015.

[I-D.wijnands-mpls-bier-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-wijnands-mpls-bier-encapsulation-02 (work in progress), December 2014.

[I-D.xu-bier-encapsulation]
Xu, X., Somasundaram, S., Jacquenet, C., and R. Raszuk, "BIER Encapsulation", draft-xu-bier-encapsulation-02 (work in progress), February 2015.

[IEEE802.1Q-2014]
IEEE, "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks", IEEE Std 802.1Q-2014, 2014, <http://www.ieee802.org/1/pages/802.1Q-2014.html>.

(Access Controlled link within page)

[RFC0033] Crocker, S., "New Host-Host Protocol", RFC 33, February 1970.

Authors' Addresses

Erik Nordmark
Arista Networks
5453 Great America Parkway
Santa Clara, CA 95054
USA

Email: nordmark@arista.com

Albert Tian
Ericsson Inc.
300 Holger Way
San Jose, California 95134
USA

Email: albert.tian@ericsson.com

Jesse Gross
VMware
3401 Hillview Ave.
Palo Alto, CA 94304
USA

Email: jgross@vmware.com

Jon Hudson
Brocade Communications Systems, Inc.
130 Holger Way
San Jose, CA 95134
USA

Email: jon.hudson@gmail.com

Lawrence Kreeger
Cisco Systems, Inc.
170 W. Tasman Avenue
San Jose, CA 95134
USA

Email: kreeger@cisco.com

Pankaj Garg
Microsoft
1 Microsoft Way
Redmond, WA 98052
USA

Email: pankajg@microsoft.com

Patricia Thaler
Broadcom Corporation
3151 Zanker Road
San Jose, CA 95134
USA

Email: pthaler@broadcom.com

Tom Herbert
Facebook
1 Hacker Way
Menlo Park, CA 94052
USA

Email: tom@herbertland.com

RTGWG
Internet-Draft
Intended status: Informational
Expires: May 4, 2017

E. Nordmark (ed)
Arista Networks
A. Tian
Ericsson Inc.
J. Gross
VMware
J. Hudson
Brocade Communications Systems, Inc.
L. Kreeger
Cisco Systems, Inc.
P. Garg
Microsoft
P. Thaler
Broadcom Corporation
T. Herbert
Facebook
October 31, 2016

Encapsulation Considerations
draft-ietf-rtgwg-dt-encap-02

Abstract

The IETF Routing Area director has chartered a design team to look at common issues for the different data plane encapsulations being discussed in the NVO3 and SFC working groups and also in the BIER BoF, and also to look at the relationship between such encapsulations in the case that they might be used at the same time. The purpose of this design team is to discover, discuss and document considerations across the different encapsulations in the different WGs/BoFs so that we can reduce the number of wheels that need to be reinvented in the future.

This Internet-Draft will expire on May 4, 2017.

Copyright Notice

Table of Contents

1. Design Team Charter

There have been multiple efforts over the years that have resulted in new or modified data plane behaviors involving encapsulations. That includes IETF efforts like MPLS, LISP, and TRILL but also industry efforts like VXLAN and NVGRE. These collectively can be seen as a source of insight into the properties that data planes need to meet. The IETF is currently working on potentially new encapsulations in NVO3 and SFC and considering working on BIER. In addition there is work on tunneling in the INT area.

This is a short term design team chartered to collect and construct useful advice to parties working on new or modified data plane behaviors that include additional encapsulations. The goal is for the group to document useful advice gathered from interacting with ongoing efforts. An Internet Draft will be produced for IETF92 to capture that advice, which will be discussed in RTGWG.

Data plane encapsulations face a set of common issues such as:

- How to provide entropy for ECMP
- Issues around packet size and fragmentation/reassembly
- OAM - what support is needed in an encapsulation format?
- Security and privacy.
- QoS
- Congestion Considerations
- IPv6 header protection (zero UDP checksum over IPv6 issue)
- Extensibility - e.g., for evolving OAM, security, and/or congestion control
- Layering of multiple encapsulations e.g., SFC over NVO3 over BIER

The design team will provide advice on those issues. The intention is that even where we have different encapsulations for different purposes carrying different information, each such encapsulation doesn't have to reinvent the wheel for the above common issues.

The design team will look across the routing area in particular at SFC, NVO3 and BIER. It will not be involved in comparing or analyzing any particular encapsulation formats proposed in those WGs and BoFs but instead focus on common advice.

2. Overview

The references provide background information on NVO3, SFC, and BIER. In particular, NVO3 is introduced in [RFC7364], [RFC7365], and [I-D.ietf-nvo3-arch]. SFC is introduced in [I-D.ietf-sfc-architecture] and [I-D.ietf-sfc-problem-statement]. Finally, the information on BIER is in [I-D.shepherd-bier-problem-statement], [I-D.wijnands-bier-architecture], and [I-D.wijnands-mpls-bier-encapsulation]. We assume the reader has some basic familiarity with those proposed encapsulations. The Related Work section points at some prior work that relates to the encapsulation considerations in this document.

Encapsulation protocols typically have some unique information that they need to carry. In some cases that information might be modified along the path and in other cases it is constant. The in-flight modifications has impacts on what it means to provide security for the encapsulation headers.

- o NVO3 carries a VNI Identifier edge to edge which is not modified. There has been OAM discussions in the WG and it isn’t clear whether some of the OAM information might be modified in flight.
- o SFC carries Service Function Path identification and service meta-data. The meta-data might be modified as the packets follow the service path. SFC talks of some loop avoidance mechanism which is likely to result in modifications for for each hop in the service chain even if the meta-data is unmodified.
- o BIER carries a bitmap of egress ports to which a packet should be delivered, and as the packet is forwarded down different paths different bits are cleared in that bitmap.

Even if information isn’t modified in flight there might be devices that wish to inspect that information. For instance, one can envision future NVO3 security devices which filter based on the virtual network identifier.

The need for extensibility is different across the protocols

- o NVO3 might need some extensions for OAM and security.
- o SFC consists of Service Function Path identification plus carrying service meta-data along a path, and different services might need different types and amount of meta-data.
- o BIER might need variable number of bits in their bitmaps, or other future schemes to scale up to larger network.

The extensibility needs and constraints might be different when considering hardware vs. software implementations of the

encapsulation headers. NIC hardware might have different constraints than switch hardware.

As the IETF designs these encapsulations the different WGs solve the issues for their own encapsulation. But there are likely to be future cases when the different encapsulations are combined in the same header. For instance, NVO3 might be a "transport" used to carry SFC between the different hops in the service chain.

Most of the issues discussed in this document are not new. The IETF and industry as specified and deployed many different encapsulation or tunneling protocols over time, ranging from simple IP-in-IP and GRE encapsulation, IPsec, pseudo-wires, session-based approached like L2TP, and the use of MPLS control and data planes. IEEE 802 has also defined layered encapsulation for Provider Backbone Bridges (PBB) and IEEE 802.1Qbp (ECMP). This document tries to leverage what we collectively have learned from that experience and summarize what would be relevant for new encapsulations like NVO3, SFC, and BIER.

3. Common Issues

[This section is mostly a repeat of the charter but with a few modifications and additions.]

Any new encapsulation protocol would need to address a large set of issues that are not central to the new information that this protocol intends to carry. The common issues explored in this document are:

- How to provide entropy for Equal Cost MultiPath (ECMP) routing
- Issues around packet size and fragmentation/reassembly
- Next header indication - each encapsulation might be able to carry different payloads
- OAM - what support is needed in an encapsulation format?
- Security and privacy
- QoS
- Congestion Considerations
- Header protection
- Extensibility - e.g., for evolving OAM, security, and/or congestion control
- Layering of multiple encapsulations e.g., SFC over NVO3 over BIER
- Importance of being friendly to hardware and software implementations

The degree to which these common issues apply to a particular encapsulation can differ based on the intended purpose of the encapsulation. But it is useful to understand all of them before determining which ones apply.

4. Scope

It is important to keep in mind what we are trying to cover and not cover in this document and effort. This is

- o A look across the three new encapsulations, while taking lots of previous work into account
- o Focus on the class of encapsulations that would run over IP/UDP. That was done to avoid being distracted by the data-plane and control-plane interaction, which is more significant for protocols that are designed to run over "transports" that maintain session or path state.
- o We later expanded the scope somewhat to consider how the encapsulations would play with MPLS "transport", which is important because SFC and BIER seem to target being independent of the underlying "transport"

However, this document and effort is NOT intended to:

- o Design some new encapsulation header to rule them all
- o Design yet another new NVO3 encapsulation header
- o Try to select the best encapsulation header
- o Evaluate any existing and proposed encapsulations

While the origin and focus of this document is the routing area and in particular NVO3, SFC, and BIER, the considerations apply to other encapsulations that are being defined in the IETF and elsewhere. There seems to be an increase in the number of encapsulations being defined to run over UDP, where there might already exist an encapsulation over IP or Ethernet. Feedback on how these considerations apply in those contexts is welcome.

5. Assumptions

The design center for the new encapsulations is a well-managed network. That network can be a datacenter network (plus datacenter interconnect) or a service provider network. Based on the existing and proposed encapsulations in those environment it is reasonable to make these assumptions:

- o The MTU is carefully managed and configured. Hence an encapsulation protocol can make the packets bigger without resulting in a requirement for fragmentation and reassembly between ingress and egress. (However, it might be useful to detecting MTU misconfigurations.)
- o In general an encapsulation needs some approach for congestion management. But the assumptions are different than for arbitrary Internet paths in that the underlay might be well-provisioned and

better policed at the edge, and due to multi-tenancy, the congestion control in the endpoints might be even less trusted than on the Internet at large.

The goal is to implement these encapsulations in hardware and software hence we can’t assume that the needs of either implementation approach can trump the needs of the other. In particular, around extensibility the needs and constraints might be quite different.

6. Terminology

The capitalized keyword MUST is used as defined in http://en.wikipedia.org/wiki/Julmust

TBD: Refer to existing documents for at least NVO3 and SFC terminology. We use at least the VNI ID in this document.

7. Entropy

In many cases the encapsulation format needs to enable ECMP in unmodified routers. Those routers might use different fields in TCP/UDP packets to do ECMP without a risk of reordering a flow. Note that the same entropy might also be used at layer 2 e.g. for Link Aggregation (LAG).

The common way to do ECMP-enabled encapsulation over IP today is to add a UDP header and to use UDP with the UDP source port carrying entropy from the inner/original packet headers as in LISP [RFC6830]. The total entropy consists of 14 bits in the UDP source port (using the ephemeral port range) plus the outer IP addresses which seems to be sufficient for entropy; using outer IPv6 headers would give the option for more entropy should it be needed in the future.

In some environments it might be fine to use all 16 bits of the port range. However, middleboxes might make assumptions about the system ports or user ports. But they should not make any assumptions about the ports in the Dynamic and/or Private Port range, which have the two MSBs set to 11b.

The UDP source port might change over the lifetime of an encapsulated flow, for instance for DoS mitigation or re-balancing load across ECMP. Such changes need to consider reordering if there are packets in flight for the flow.

There is some interaction between entropy and OAM and extensibility mechanism. It is desirable to be able to send OAM packets to follow the same path as network packets. Hence OAM packets should use the

same entropy mechanism as data packets. While routers might use information in addition the entropy field and outer IP header, they can not use arbitrary parts of the encapsulation header since that might result in OAM frames taking a different path. Likewise if routers look past the encapsulation header they need to be aware of the extensibility mechanism(s) in the encapsulation format to be able to find the inner headers in the presence of extensions; OAM frames might use some extensions e.g. for timestamps.

Architecturally the entropy and the next header field are really part of enclosing delivery header. UDP with entropy goes hand-in-hand with the outer IP header. Thus the UDP entropy is present for the underlay IP routers the same way that an MPLS entropy label is present for LSRs. The entropy above is all about providing entropy for the outer delivery of the encapsulated packets.

It has been suggested that when IPv6 is used it would not be necessary to add a UDP header for entropy, since the IPv6 flow label can be used for entropy. (This assumes that there is an IP protocol number for the encapsulation in addition to a UDP destination port number since UDP would be used with IPv4 underlay. And any use of UDP checksums would need to be replaced by an encaps-specific checksum or secure hash.) While such an approach would save 8 bytes of headers when the underlay is IPv6, it does assume that the underlay routers use the flow label for ECMP, and it also would make the IPv6 approach different than the IPv4 approach. Currently the leaning is towards recommending using the UDP encapsulation for both IPv4 and IPv6 underlay. The IPv6 flow label can be used for additional entropy if need be. There is more detailed discussion for using the IPv6 flow label for tunnels in [RFC6438].

Note that in the proposed BIER encapsulation [I-D.wijnands-mpls-bier-encapsulation], there is an an 8-bit field which specifies an entropy value that can be used for load balancing purposes. This entropy is for the BIER forwarding decisions, which is independent of any outer delivery ECMP between BIER routers. Thus it is not part of the delivery ECMP discussed in this section.

> [Note: For any given bit in BIER (that identifies an exit from the BIER domain) there might be multiple immediate next hops. The BIER entropy field is used to select that next hop as part of BIER processing. The BIER forwarding process may do equal cost load balancing, but the load balancing procedure MUST choose the same path for any two packets that have the same entropy value.]

In summary:

- o The entropy is associated with the transport, that is an outer IP header or MPLS.
- o In the case of IP transport use 14 or 16 bits of UDP source port, plus outer IPv6 flowid for entropy.

8. Next-protocol indication

Next-protocol indications appear in three different contexts for encapsulations.

Firstly, the transport delivery mechanism for the encapsulations we discuss in this document need some way to indicate which encapsulation header (or other payload) comes next in the packet. Some encapsulations might be identified by a UDP port; others might be identified by an Ethernet type or IP protocol number. Which approach is used is a function of the preceding header the same way as IPv4 is identified by both an Ethernet type and an IP protocol number (for IP-in-IP). In some cases the header type is implicit in some session (L2TP) or path (MPLS) setup. But this is largely beyond the control of the encapsulation protocol. For instance, if there is a requirement to carry the encapsulation after an Ethernet header, then an Ethernet type is needed. If required to be carried after an IP/UDP header, then a UDP port number is needed. For UDP port numbers there are considerations for port number conservation described in [I-D.ietf-tsvwg-port-use].

It is worth mentioning that in the MPLS case of no implicit protocol type many forwarding devices peek at the first nibble of the payload to determine whether to apply IPv4 or IPv6 L3/L4 hashes for load balancing [RFC7325]. That behavior places some constraints on other payloads carried over MPLS and some protocol define an initial control word in the payload with a value of zero in its first nibble [RFC4385] to avoid confusion with IPv4 and IPv6 payload headers.

Secondly, the encapsulation needs to indicate the type of its payload, which is in scope for the design of the encapsulation. We have existing protocols which use Ethernet types (such as GRE). Here each encapsulation header can potentially makes its own choices between:

- o Use the Ethernet type space - makes it easy to carry existing L2 and L3 protocols including IPv4, IPv6, and Ethernet. Disadvantages are that it is a 16 bit number and we probably need far less than 100 values, and the number space is controlled by the IEEE 802 RAC with its own allocation policies.
- o Use the IP protocol number space - makes it easy to carry e.g., ESP in addition to IP and Ethernet but brings in all existing protocol numbers many of which would never be used directly on top

of the encapsulation protocol. IANA managed eight bit values, presumably more difficult to get an assigned number than to get a transport port assignment.

- o Define their own next-protocol number space, which can use fewer bits than an Ethernet type and give more flexibility, but at the cost of administering that numbering space (presumably by the IANA).

Thirdly, if the IETF ends up defining multiple encapsulations at about the same time, and there is some chance that multiple such encapsulations can be combined in the same packet, there is a question whether it makes sense to use a common approach and numbering space for the encapsulation across the different protocols. A common approach might not be beneficial as long as there is only one way to indicate e.g., SFC inside NVO3.

Many Internet protocols use fixed values (typically managed by the IANA function) for their next-protocol field. That facilitates interpretation of packets by middleboxes and e.g., for debugging purposes, but might make the protocol evolution inflexible. Our collective experience with MPLS shows an alternative where the label can be viewed as an index to a table containing processing instructions and the table content can be managed in different ways. Encapsulations might want to consider the tradeoffs between such more flexible versus more fixed approaches.

In summary:

- o Would it be useful for the IETF come up with a common scheme for encapsulation protocols? If not each encapsulation can define its own scheme.

9. MTU and Fragmentation

A common approach today is to assume that the underlay have sufficient MTU to carry the encapsulated packets without any fragmentation and reassembly at the tunnel endpoints. That is sufficient when the operator of the ingress and egress have full control of the paths between those endpoints. And it makes for simpler (hardware) implementations if fragmentation and reassembly can be avoided.

However, even under that assumption it would be beneficial to be able to detect when there is some misconfiguration causing packets to be dropped due to MTU issues. One way to do this is to have the encapsulator set the don’t-fragment (DF) flag in the outer IPv4 header and receive and log any received ICMP "packet too big" (PTB)

errors. Note that no flag needs to be set in an outer IPv6 header [RFC2460].

Encapsulations could also define an optional tunnel fragmentation and reassembly mechanism which would be useful in the case when the operator doesn’t have full control of the path, or when the protocol gets deployed outside of its original intended context. Such a mechanism would be required if the underlay might have a path MTU which makes it impossible to carry at least 1518 bytes (if offering Ethernet service), or at least 1280 (if offering IPv6 service). The use of such a protocol mechanism could be triggered by receiving a PTB. But such a mechanism might not be implemented by all encapsulators and decapsulators. [Aerolink is one example of such a protocol.]

Depending on the payload carried by the encapsulation there are some additional possibilities:

- o If payload is IPv4/6 then the underlay path MTU could be used to report end-to-end path MTU.
- o If the payload service is Ethernet/L2, then there is no such per destination reporting mechanism. However, there is a LLDP TLV for reporting max frame size; might be useful to report minimum to end stations, but unmodified end stations would do nothing with that TLV since they assume that the MTU is at least 1518.

In summary:

- o In some deployments an encapsulation can assume well-managed MTU hence no need for fragmentation and reassembly related to the encapsulation.
- o Even so, it makes sense for ingress to track any ICMP packet too big addressed to ingress to be able to log any MTU misconfigurations.
- o Should an encapsulation protocol be deployed outside of the original context it might very well need support for fragmentation and reassembly.

10. OAM

The OAM area is seeing active development in the IETF with discussions (at least) in NVO3 and SFC working groups, plus the new LIME WG looking at architecture and YANG models.

The design team has take a narrow view of OAM to explore the potential OAM implications on the encapsulation format.

In terms of what we have heard from the various working groups there seem to be needs to:

- o Be able to send out-of-band OAM messages - that potentially should follow the same path through the network as some flow of data packets.
  - * Such OAM messages should not accidentally be decapsulated and forwarded to the end stations.
- o Be able to add OAM information to data packets that are encapsulated. Discussions have been around:
  - * Using a bit in the OAM to synchronize sampling of counters between the encapsulator and decapsulator.
  - * Optional timestamps, sequence numbers, etc for more detailed measurements between encapsulator and decapsulator.
- o Usable for both proactive monitoring (akin to BFD) and reactive checks (akin to traceroute to pin-point a failure)

To ensure that the OAM messages can follow the same path the OAM messages need to get the same ECMP (and LAG hashing) results as a given data flow. An encapsulator can choose between one of:

- o Limit ECMP hashing to not look past the UDP header i.e. the entropy needs to be in the source/destination IP and UDP ports
- o Make OAM packets look the same as data packets i.e. the initial part of the OAM payload has the inner Ethernet, IP, TCP/UDP headers as a payload. (This approach was taken in TRILL out of necessity since there is no UDP header.) Any OAM bit in the encapsulation header must in any case be excluded from the entropy.

There can be several ways to prevent OAM packets from accidentally being forwarded to the end station using:

- o A bit in the frame (as in TRILL) indicating OAM
- o A next-protocol indication with a designated value for "none" or "oam".

This assumes that the bit or next protocol, respectively, would not affect entropy/ECMP in the underlay. However, the next-protocol field might be used to provide differentiated treatment of packets based on their payload; for instance a TCP vs. IPsec ESP payload might be handled differently. Based on that observation it might be undesirable to overload the next protocol with the OAM drop behavior, resulting in a preference for having a bit to indicate that the packet should be forwarded to the end station after decapsulation.

There has been suggestions that one (or more) marker bits in the encaps header would be useful in order to delineate measurement epochs on the encapsulator and decapsulator and use that to compare counters to determine packet loss.

A result of the above is that OAM is likely to evolve and needs some degree of extensibility from the encapsulation format; a bit or two plus the ability to define additional larger extensions.

An open question is how to handle error messages or other reports relating to OAM. One can think if such reporting as being associated with the encapsulation the same way ICMP is associated with IP. Would it make sense for the IETF to develop a common Encapsulation Error Reporting Protocol as part of OAM, which can be used for different encapsulations? And if so, what are the technical challenges. For instance, how to avoid it being filtered as ICMP often is?

A potential additional consideration for OAM is the possible future existence of gateways that "stitch" together different dataplane encapsulations and might want to carry OAM end-to-end across the different encapsulations.

In summary:

- It makes sense to reserve a bit for "drop after decapsulation" for OAM out-of-band.
- An encapsulation needs sufficient extensibility for OAM (such as bits, timestamps, sequence numbers). That might be motivated by in-band OAM but it would make sense to leverage the same extensions for out-of band OAM.
- OAM places some constraints on use of entropy in forwarding devices.
- Should IETF look into error reporting that is independent of the specific encapsulation?

11. Security Considerations

Different encapsulation use cases will have different requirements around security. For instance, when encapsulation is used to build overlay networks for network virtualization, isolation between virtual networks may be paramount. BIER support of multicast may entail different security requirements than encapsulation for unicast.

In real deployment, the security of the underlying network may be considered for determining the level of security needed in the encapsulation layer. However for the purposes of this discussion, we

assume that network security is out of scope and that the underlying network does not itself provide adequate or as least uniform security mechanisms for encapsulation.

There are at least three considerations for security:

- o Anti-spoofing/virtual network isolation
- o Interaction with packet level security such as IPsec or DTLS
- o Privacy (e.g., VNI ID confidentially for NVO3)

This section uses a VNI ID in NVO3 as an example. A SFC or BIER encapsulation is likely to have fields with similar security and privacy requirements.

11.1. Encapsulation-specific considerations

Some of these considerations appear for a new encapsulation, and others are more specific to network virtualization in datacenters.

- o New attack vectors:
  - * DDOS on specific queued/paths by attempting to reproduce the 5-tuple hash for targeted connections.
  - * Entropy in outer 5-tuple may be too little or predictable.
  - * Leakage of identifying information in the encapsulation header for an encrypted payload.
  - * Vulnerabilities of using global values in fields like VNI ID.
- o Trusted versus untrusted tenants in network virtualization:
  - * The criticality of virtual network isolation depends on whether tenants are trusted or untrusted. In the most extreme cases, tenants might not only be untrusted but may be considered hostile.
  - * For a trusted set of users (e.g. a private cloud) it may be sufficient to have just a virtual network identifier to provide isolation. Packets inadvertently crossing virtual networks should be dropped similar to a TCP packet with a corrupted port being received on the wrong connection.
  - * In the presence of untrusted users (e.g. a public cloud) the virtual network identifier must be adequately protected against corruption and verified for integrity. This case may warrant keyed integrity.
- o Different forms of isolation:
  - * Isolation could be blocking all traffic between tenants (or except as allowed by some firewall)
  - * Could also be about performance isolation i.e. one tenant can overload the network in a way that affects other tenants

  * Physical isolation of traffic for different tenants in network may be required, as well as required restrictions that tenants may have on where their packets may be routed.
- New attack vectors from untrusted tenants:
  * Third party VMs with untrusted tenants allows internally borne attacks within data centers
  * Hostile VMs inside the system may exist (e.g. public cloud)
  * Internally launched DDOS
  * Passive snooping for mis-delivered packets
  * Mitigate damage and detection in event that a VM is able to circumvent isolation mechanisms
- Tenant-provider relationship:
  * Tenant might not trust provider, hypervisors, network
  * Provider likely will need to provide SLA or a least a statement on security
  * Tenant may implement their own additional layers of security
  * Regulation and certification considerations
- Trend towards tighter security:
  * Tenants' data in network increases in volume and value, attacks become more sophisticated
  * Large DCs already encrypt everything on disk
  * DCs likely to encrypt inter-DC traffic at this point, use TLS to Internet.
  * Encryption within DC is becoming more commonplace, becomes ubiquitous when cost is low enough.
  * Cost/performance considerations. Cost of support for strong security has made strong network security in DCs prohibitive.
  * Are there lessons from MacSec?

11.2. Virtual network isolation

The first requirement is isolation between virtual networks. Packets sent in one virtual network should never be illegitimately received by a node in another virtual network. Isolation should be protected in the presence of malicious attacks or inadvertent packet corruption.

The second requirement is sender authentication. Sender identity is authenticated to prevent anti-spoofing. Even if an attacker has access to the packets in the network, they cannot send packets into a virtual network. This may have two possibilities:

- Pairwise sender authentication. Any two communicating hosts negotiate a shared key.

- o Group authentication. A group of hosts share a key (this may be more appropriate for multicast of encapsulation).

Possible security solutions:

- o Security cookie: This is similar to L2TP cookie mechanism [RFC3931]. A shared plain text cookie is shared between encapsulator and decapsulator. A receiver validates a packet by evaluating if the cookie is correct for the virtual network and address of a sender. Validation function is F(cookie, VNI ID, source address). If cookie matches, accept packet, else drop. Since cookie is plain text this method does not protect against an eavesdropping. Cookies are set and may be rotated out of band.
- o Secure hash: This is a stronger mechanism than simple cookies that borrows from IPsec and PPP authentication methods. In this model security field contains a secure hash of some fields in the packet using a shared key. Hash function may be something like H(key, VNI ID, address, salt). The salt ensures the hash is not the same for every packet, and if it includes a sequence number may also protect against replay attacks.

In any use of a shared key, periodic re-keying should be allowed. This could include use of techniques like generation numbers, key windows, etc. See [I-D.farrelll-mpls-opportunistic-encrypt] for an example application.

We might see firewalls that are aware of the encapsulation and can provide some defense in depth combined with the above example anti-spoofing approaches. An example would be an NVO3-aware firewall being able to check the VNI ID.

Separately and in addition to such filtering, there might be a desire to completely block an encapsulation protocol at certain places in the network, e.g., at the edge of a datacenter. Using a fixed standard UDP destination port number for each encapsulation protocol would facilitate such blocking.

11.3. Packet level security

An encapsulated packet may itself be encapsulated in IPsec (e.g. ESP). This should be straightforward and in fact is what would happen today in security gateways. In this case, there is no special consideration for the fact that packet is encapsulated, however since the encapsulation layer headers are included (part of encrypted data for instance) we lose visibility in the network of the encapsulation.

The more interesting case is when security is applied to the encapsulation payload. This will keep the encapsulation headers in

the outer header visible to the network (for instance in nvo3 we may way to firewall based on VNI ID even if the payload is encrypted). One possibility is to apply DTLS to the encapsulation payload. In this model the protocol stack may be something like IP|UDP|Encap|DTLS|encrypted_payload. The encapsulation and security should be done together at an encapsulator and resolved at the decapsulator. Since the encapsulation header is outside of the security coverage, this may itself require security (like described above).

In both of the above the security associations (SAs) may be between physical hosts, so for instance in nvo3 we can have packets of different virtual networks using the same SA-- this should not be an issue since it is the VNI ID that ensures isolation (which needs to be secured also).

11.4. In summary:

- o Encapsulations need extensibility mechanisms to be able to add security features like cookies and secure hashes protecting the encapsulation header.
- o NVO3 probably has specific higher requirements relating to isolation for network virtualization, which is in scope for the NVO3 WG.
- o Our collective IETF experience is that successful protocols get deployed outside of the original intended context, hence the initial assumptions about the threat model might become invalid. That needs to be considered in the standardization of new encapsulations.

12. QoS

In the Internet architecture we support QoS using the Differentiated Services Code Points (DSCP) in the formerly named Type-of-Service field in the IPv4 header, and in the Traffic-Class field in the IPv6 header. The ToS and TC fields also contain the two ECN bits, which are discussed in Section 13.

We have existing specifications how to process those bits. See [RFC2983] for diffserv handling, which specifies how the received DSCP value is used to set the DSCP value in an outer IP header when encapsulating. (There are also existing specifications how DSCP can be mapped to layer2 priorities.)

Those specifications apply whether or not there is some intervening headers (e.g., for NVO3 or SFC) between the inner and outer IP headers. Thus the encapsulation considerations in this area are mainly about applying the framework in [RFC2983].

Note that the DSCP and ECN bits are not the only part of an inner packet that might potentially affect the outer packet. For example, [RFC2473] specifies handling of inner IPv6 hop-by-hop options that effectively result in copying some options to the outer header. It is simpler to not have future encapsulations depend on such copying behavior.

There are some other considerations specific to doing OAM for encapsulations. If OAM messages are used to measure latency, it would make sense to treat them the same as data payloads. Thus they need to have the same outer DSCP value as the data packets which they wish to measure.

Due to OAM there are constraints on middleboxes in general. If middleboxes inspect the packet past the outer IP+UDP and encapsulation header and look for inner IP and TCP/UDP headers, that might violate the assumption that OAM packets will be handled the same as regular data packets. That issue is broader than just QoS - applies to firewall filters etc.

In summary:

o Leverage the existing approach in [RFC2983] for DSCP handling.

13. Congestion Considerations

Additional encapsulation headers does not introduce anything new for Explicit Congestion Notification. It is just like IP-in-IP and IPsec tunnels which is specified in [RFC6040] in terms of how the ECN bits in the inner and outer header are handled when encapsulating and decapsulating packets. Thus new encapsulations can more or less include that by reference.

There are additional considerations around carrying non-congestion controlled traffic. These details have been worked out in [I-D.ietf-mpls-in-udp]. As specified in [RFC5405]: "IP-based traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. Consequently, a tunnel carrying IP-based traffic should already interact appropriately with other traffic sharing the path, and specific congestion control mechanisms for the tunnel are not necessary". Those considerations are being captured in [I-D.ietf-tsvwg-rfc5405bis].

For this reason, where an encapsulation method is used to carry IP traffic that is known to be congestion controlled, the UDP tunnels does not create an additional need for congestion control. Internet

IP traffic is generally assumed to be congestion-controlled. Similarly, in general Layer 3 VPNs are carrying IP traffic that is similarly assumed to be congestion controlled.

However, some of the encapsulations (at least NVO3) will be able to carry arbitrary Layer 2 packets to provide an L2 service, in which case one can not assume that the traffic is congestion controlled.

One could handle this by adding some congestion control support to the encapsulation header (one instance of which would end up looking like DCCP). However, if the underlay is well-provisioned and managed as opposed to being arbitrary Internet path, it might be sufficient to have a slower reaction to congestion induced by that traffic. There is work underway on a notion of "circuit breakers" for this purpose. See See [I-D.ietf-tsvwg-circuit-breaker]. Encapsulations which carry arbitrary Layer 2 packets want to consider that ongoing work.

If the underlay is provisioned in such a way that it can guarantee sufficient capacity for non-congestion controlled Layer 2 traffic, then such circuit breakers might not be needed.

Two other considerations appear in the context of these encapsulations as applied to overlay networks:

- o Protect against malicious end stations
- o Ensure fairness and/or measure resource usage across multiple tenants

Those issues are really orthogonal to the encapsulation, in that they are present even when no new encapsulation header is in use. However, the application of the new encapsulations are likely to be in environments where those issues are becoming more important. Hence it makes sense to consider them.

One could make the encapsulation header be extensible to that it can carry sufficient information to be able to measure resource usage, delays, and congestion. The suggestions in the OAM section about a single bit for counter synchronization, and optional timestamps and/or sequence numbers, could be part of such an approach. There might also be additional congestion-control extensions to be carried in the encapsulation. Overall this results in a consideration to support sufficient extensibility in the encapsulation to handle potential future developments in this space.

Coarse measurements are likely to suffice, at least for circuit-breaker-like purposes, see [I-D.wei-tsvwg-tunnel-congestion-feedback] and [I-D.briscoe-conex-data-centre] for examples on active work in

this area via use of ECN. [RFC6040] Appendix C is also relevant. The outer ECN bits seem sufficient (at least when everything uses ECN) to do this course measurements. Needs some more study for the case when there are also drops; might need to exchange counters between ingress and egress to handle drops.

Circuit breakers are not sufficient to make a network with different congestion control when the goal is to provide a predictable service to different tenants. The fallback would be to rate limit different traffic.

In summary:

- o Leverage the existing approach in [RFC6040] for ECN handling.
- o If the encapsulation can carry non-IP, hence non-congestion controlled traffic, then leverage the approach in [I-D.ietf-mpls-in-udp].
- o "Watch this space" for circuit breakers.

14. Header Protection

Many UDP based encapsulations such as VXLAN [RFC7348] either discourage or explicitly disallow the use of UDP checksums. The reason is that the UDP checksum covers the entire payload of the packet and switching ASICs are typically optimized to look at only a small set of headers as the packet passes through the switch. In these case, computing a checksum over the packet is very expensive. (Software endpoints and the NICs used with them generally do not have the same issue as they need to look at the entire packet anyways.)

The lack a header checksum creates the possibility that bit errors can be introduced into any information carried by the new headers. Specifically, in the case of IPv6, the assumption is that a transport layer checksum - UDP in this case - will protect the IP addresses through the inclusion of a pseudo-header in the calculation. This is different from IPv4 on which many of these encapsulation protocols are initially deployed which contains its own header checksum. In addition to IP addresses, the encapsulation header often contains its own information which is used for addressing packets or other high value network functions. Without a checksum, this information is potentially vulnerable - an issue regardless of whether the packet is carried over IPv4 or IPv6.

Several protocols cite [RFC6935] and [RFC6936] as an exemption to the IPv6 checksum requirements. However, these are intended to be tailored to a fairly narrow set of circumstances - primarily relying on sparseness of the address space to detect invalid values and well managed networks - and are not a one size fits all solution. In

these cases, an analysis should be performed of the intended environment, including the probability of errors being introduced and the use of ECC memory in routing equipment.

Conceptually, the ideal solution to this problem is a checksum that covers only the newly added headers of interest. There is little value in the portion of the UDP checksum that covers the encapsulated packet because that would generally be protected by other checksums and this is the expensive portion to compute. In fact, this solution already exists in the form of UDP-Lite and UDP based encapsulations could be easily ported to run on top of it. Unfortunately, the main value in using UDP as part of the encapsulation header is that it is recognized by already deployed equipment for the purposes of ECMP, RSS, and middlebox operations. As UDP-Lite uses a different protocol number than UDP and it is not widely implemented in middleboxes, this value is lost. A possible solution is to incorporate the same partial-checksum concept as UDP-Lite or other header checksum protection into the encapsulation header and continue using UDP as the outer protocol. One potential challenge with this approach is the use of NAT or other form of translation on the outer header will result in an invalid checksum as the translator will not know to update the encapsulation header.

The method chosen to protect headers is often related to the security needs of the encapsulation mechanism. On one hand, the impact of a poorly protected header is not limited to only data corruption but can also introduce a security vulnerability in the form of misdirected packets to an unauthorized recipient. Conversely, high security protocols that already include a secure hash over the valuable portion of the header (such as by encrypting the entire IP packet using IPsec, or some secure hash of the encap header) do not require additional checksum protection as the hash provides stronger assurance than a simple checksum.

If the sender has included a checksum, then the receiver should verify that checksum or, if incapable, drop the packet. The assumption is that configuration and/or control-plane capability exchanges can be used when different receiver have different checksum validation capabilities.

In summary:

- o Encapsulations need extensibility to be able to add checksum/CRC for the encapsulation header itself.
- o When the encapsulation has a checksum/CRC, include the IPv6 pseudo-header in it.
- o The checksum/CRC can potentially be avoided when cryptographic protection is applied to the encapsulation.

15. Extensibility Considerations

Protocol extensibility is the concept that a networking protocol may be extended to include new use cases or functionality that were not part of the original protocol specification. Extensibility may be used to add security, control, management, or performance features to a protocol. A solution may allow private extensions for customization or experimentation.

Extending a protocol often implies that a protocol header must carry new information. There are two usual methods to accomplish this:

1. Define or redefine the meaning of existing fields in a protocol header.
2. Add new (optional) fields to the protocol header.

It is also possible to create a new protocol version, but this is more associated with defining a protocol than extending it (IPv6 being a successor to IPv4 is an example of protocol versioning).

In some cases it might be more appropriate to define a new inner protocol which can carry the new functionality instead of extending the outer protocol. Examples where this works well is in the IP/transport split, where the earlier architecture had a single NCP [RFC0033] protocol which carried both the hop-by-hop semantics which are now in IP, and the end-to-end semantics which are now in TCP. Such a split is effective when different nodes need to act upon the different information. Applying this for general protocol extensibility through nesting is not well understood, and does result in longer header chains. Furthermore, our experience with IPv6 extension headers [RFC2460] in middleboxes indicates that the header chaining approach does not help with middlebox traversal.

Many protocol definitions include some number of reserved fields or bits which can be used for future extension. VXLAN is an example of a protocol that includes reserved bits which are subsequently being allocated for new purposes. Another technique employed is to re-purpose existing header fields with new meanings. A classic example of this is the definition of DSCP code point which redefines the ToS field originally specified in IPv4. When a field is redefined, some mechanism may be needed to ensure that all interested parties agree on the meaning of the field. The techniques of defining meaning for reserved bits or redefining existing fields have the advantage that a protocol header can be kept a fixed length. The disadvantage is that the extensibility is limited. For instance, the number reserved bits in a fixed protocol header is limited. For standard protocols the decision to commit to a definition for a field can be wrenching since it is difficult to retract later. Also, it is difficult to predict a

priori how many reserved fields or bits to put into a protocol header to satisfy the extensions create over the lifetime of the protocol.

Extending a protocol header with new fields can be done in several ways.

- o TLVs are a very popular method used in such protocols as IP and TCP. Depending on the type field size and structure, TLVs can offer a virtually unlimited range of extensions. A disadvantage of TLVs is that processing them can be verbose, quite complicated, several validations must often be done for each TLV, and there is no deterministic ordering for a list of TLVs. TCP serves as an example of a protocol where TLVs have been successfully used (i.e. required for protocol operation). IP is an example of a protocol that allows TLVs but are rarely used in practice (router fast paths usually that assume no IP options). Note that TCP TLVs are implemented in software as well as (NIC) hardware handling various forms of TCP offload. Additional discussions about hardware implications for extensibility is captured in Section 18.
- o Extension headers are closely related to TLVs. These also carry type/value information, but instead of being a list of TLVs within a single protocol header, each one is in its own protocol header. IPv6 extension headers and SFC NSH are examples of this technique. Similar to TLVs these offer a wide range of extensibility, but have similarly complex processing. Another difference with TLVs is that each extension header is idempotent. This is beneficial in cases where a protocol implements a push/pop model for header elements like service chaining, but makes it more difficult group correlated information within one protocol header.
- o A particular form of extension headers are the tags used by IEEE 802 protocols. Those are similar to e.g., IPv6 extension headers but with the key difference that each tag is a fixed length header where the length is implicit in the tag value. Thus as long as a receiver can be programmed with a tag value to length map, it can skip those new tags.
- o Flag-fields are a non-TLV like method of extending a protocol header. The basic idea is that the header contains a set of flags, where each set flags corresponds to optional field that is present in the header. GRE is an example of a protocol that employs this mechanism. The fields are present in the header in the order of the flags, and the length of each field is fixed. Flag-fields are simpler to process compared to TLVs, having fewer validations and the order of the optional fields is deterministic. A disadvantage is that range of possible extensions with flag-fields is smaller than TLVs.

The requirements for receiving unknown or unimplemented extensible elements in an encapsulation protocol (flags, TLVs, optional fields)

need to be specified. There are two parties to consider, middle boxes and terminal endpoints of encapsulation (at the decapsulator).

A protocol may allow or expect nodes in a path to modify fields in an encapsulation (example use of this is BIER). In this case, the middleboxes should follow the same requirements as nodes terminating the encapsulation. In the case that middle boxes do not modify the encapsulation, we can assume that they may still inspect any fields of the encapsulation. Missing or unknown fields should be accepted per protocol specification, however it is permissible for a site to implement a local policy otherwise (e.g. a firewall may drop packets with unknown options).

For handling unknown options at terminal nodes, there are two possibilities: drop packet or accept while ignoring the unknown options. Many Internet protocols specify that reserved flags must be set to zero on transmission and ignored on reception. L2TP is example data protocol that has such flags. GRE is a notable exception to this rule, reserved flag bits 1-5 cannot be ignored [RFC2890]. For TCP and IPv4, implementations must ignore optional TLVs with unknown type; however in IPv6 if a packet contains an unknown extension header (unrecognized next header type) the packet must be dropped with an ICMP error message returned. The IPv6 options themselves (encoded inside the destinations options or hop-by-hop options extension header) have more flexibility. There are bits in the option code are used to instruct the receiver whether to ignore, silently drop, or drop and send error if the option is unknown. Some protocols define a "mandatory bit" that can is set with TLVs to indicate that an option must not be ignored. Conceptually, optional data elements can only be ignored if they are idempotent and do not alter how the rest of the packet is parsed or processed.

Depending on what type of protocol evolution one can predict, it might make sense to have a way for a sender to express that the packet should be dropped by a terminal node which does not understand the new information. In other cases it would make sense to have the receiver silently ignore the new info. The former can be expressed by having a version field in the encapsulation, or a notion of "mandatory bit" as discussed above.

A security mechanism which use some form secure hash over the encapsulation header would need to be able to know which extensions can be changed in flight.

In summary:

- o Encapsulations need the ability to be extended to handle e.g., the OAM or security aspects discussed in this document.
- o Practical experience seems to tell us that extensibility mechanisms which are not in use on day one might result in immediate ossification by lack of implementation support. In some cases that has occurred in routers and in other cases in middleboxes. Hence devising ways where the extensibility mechanisms are in use seems important.

16. Layering Considerations

One can envision that SFC might use NVO3 as a delivery/transport mechanism. With more imagination that in turn might be delivered using BIER. Thus it is useful to think about what things look like when we have BIER+NVO3+SFC+payload. Also, if NVO3 is widely deployed there might be cases of NVO3 nesting where a customer uses NVO3 to provide network virtualization e.g., across departments. That customer uses a service provider which happens to use NVO3 to provide transport for their customers.Thus NVO3 in NVO3 might happen.

A key question we set out to answer is what the packets might look like in such a case, and in particular whether we would end up with multiple UDP headers for entropy.

Based on the discussion in the Entropy section, the entropy is associated with the outer delivery IP header. Thus if there are multiple IP headers there would be a UDP header for each one of the IP headers. But SFC does not require its own IP header. So a case of NVO3+SFC would be IP+UDP+NVO3+SFC. A nested NVO3 encapsulation would have independent IP+UDP headers.

The layering also has some implications for middleboxes.

- o A device on the path between the ingress and egress is allowed to transparently inspect all layers of the protocol stack and drop or forward, but not transparently modify anything but the layer in which they operate. What this means is that an IP router is allowed modify the outer IP ttl and ECN bits, but not the encapsulation header or inner headers and payload. And a BIER router is allowed to modify the BIER header.
- o Alternatively such a device can become visible at a higher layer. E.g., a middlebox could a middlebox could first decapsulate, perform some function then encapsulate; which means it will generate a new encapsulation header.

The design team asked itself some additional questions:

- o Would it make sense to have a common encapsulation base header (for OAM, security?, etc) and then followed by the specific information for NVO3, SFC, BIER? Given that there are separate proposals and the set of information needing to be carried differs, and the extensibility needs might be different, it would be difficult and not that useful to have a common base header.
- o With a base header in place, one could view the different functions (NVO3, SFC, and BIER) as different extensions to that base header resulting in encodings which are more space optimal by not repeating the same base header. The base header would only be repeated when there is an additional IP (and hence UDP) header. That could mean a single length field (to skip to get to the payload after all the encapsulation headers). That might be technically feasible, but it would create a lot of dependencies between different WGs making it harder to make progress. Compare with the potential savings in packet size.

17. Service model

The IP service is lossy and subject to reordering. In order to avoid a performance impact on transports like TCP the handling of packets is designed to avoid reordering packets that are in the same transport flow (which is typically identified by the 5-tuple). But across such flows the receiver can see different ordering for a given sender. That is the case for a unicast vs. a multicast flow from the same sender.

There is a general tussle between the desire for high capacity utilization across a multipath network and the impact on packet ordering within the same flow (which results in lower transport protocol performance). That isn’t affected by the introduction of an encapsulation. However, the encapsulation comes with some entropy, and there might be cases where folks want to change that in response to overload or failures. For instance, one might want to change UDP source port to try different ECMP route. Such changes can result in packet reordering within a flow, hence would need to be done infrequently and with care e.g., by identifying packet trains.

There might be some applications/services which are not able to handle reordering across flows. The IETF has defined pseudo-wires [RFC3985] which provides the ability to ensure ordering (implemented using sequence numbers and/or timestamps).

Architectural such services would make sense, but as a separate layer on top of an encapsulation protocol. They could be deployed between ingress and egress of a tunnel which uses some encaps. Potentially the tunnel control points at the ingress and egress could become a platform for fixing suboptimal behavior elsewhere in the network.

That would clearly be undesirable in the general case. However, handling encapsulation of non-IP traffic hence non-congestion-controlled traffic is likely to be required, which implies some fairness and/or QoS policing on the ingress and egress devices.

But the tunnels could potentially do more like increase reliability (retransmissions, FEC) or load spreading using e.g. MP-TCP between ingress and egress.

18. Hardware Friendly

Hosts, switches and routers often leverage capabilities in the hardware to accelerate packet encapsulation, decapsulation and forwarding.

Some design considerations in encapsulation that leverage these hardware capabilities may result in more efficiently packet processing and higher overall protocol throughput.

While "hardware friendliness" can be viewed as unnecessary considerations for a design, part of the motivation for considering this is ease of deployment; being able to leverage existing NIC and switch chips for at least a useful subset of the functionality that the new encapsulation provides. The other part is the ease of implementing new NICs and switch/router chips that support the encapsulation at ever increasing line rates.

[disclaimer] There are many different types of hardware in any given network, each maybe better at some tasks while worse at others. We would still recommend protocol designers to examine the specific hardware that are likely to be used in their networks and make decisions on a case by case basis.

Some considerations are:

- o Keep the encap header small. Switches and routers usually only read the first small number of bytes into the fast memory for quick processing and easy manipulation. The bulk of the packets are usually stored in slow memory. A big encap header may not fit and additional read from the slow memory will hurt the overall performance and throughput.
- o Put important information at the beginning of the encapsulation header. The reasoning is similar as explained in the previous point. If important information are located at the beginning of the encapsulation header, the packet may be processed with smaller number of bytes to be read into the fast memory and improve performance.

- o Avoid full packet checksums in the encapsulation if possible. Encapsulations should instead consider adding their own checksum which covers the encapsulation header and any IPv6 pseudo-header. The motivation is that most of the switch/router hardware make switching/forwarding decisions by reading and examining only the first certain number of bytes in the packet. Most of the body of the packet do not need to be processed normally. If we are concerned of preventing packet to be misdelivered due to memory errors, consider only perform header checksums. Note that NIC chips can typically already do full packet checksums for TCP/UDP, while adding a header checksum might require adding some hardware support.
- o Place important information at fixed offset in the encapsulation header. Packet processing hardware may be capable of parallel processing. If important information can be found at fixed offset, different part of the encapsulation header may be processed by different hardware units in parallel (for example multiple table lookups may be launched in parallel). It is easier for hardware to handle optional information when the information, if present, can be found in ideally one place, but in general, in as few places as possible. That facilitates parallel processing. TLV encoding with unconstrained order typically does not have that property.
- o Limit the number of header combinations. In many cases the hardware can explore different combinations of headers in parallel, however there is some added cost for this.

18.1. Considerations for NIC offload

This section provides guidelines to provide support of common offloads for encapsulation in Network Interface Cards (NICs). Offload mechanisms are techniques that are implemented separately from the normal protocol implementation of a host networking stack and are intended to optimize or speed up protocol processing. Hardware offload is performed within a NIC device on behalf of a host.

There are three basic offload techniques of interest:

- o Receive multi queue
- o Checksum offload
- o Segmentation offload

18.1.1. Receive multi-queue

Contemporary NICs support multiple receive descriptor queues (multi-queue). Multi-queue enables load balancing of network processing for a NIC across multiple CPUs. On packet reception, a NIC must select

the appropriate queue for host processing. Receive Side Scaling (RSS) is a common method which uses the flow hash for a packet to index an indirection table where each entry stores a queue number.

UDP encapsulation, where the source port is used for entropy, should be compatible with multi-queue NICs that support five-tuple hash calculation for UDP/IP packets as input to RSS. The source port ensures classification of the encapsulated flow even in the case that the outer source and destination addresses are the same for all flows (e.g. all flows are going over a single tunnel).

18.1.2. Checksum offload

Many NICs provide capabilities to calculate standard ones complement payload checksum for packets in transmit or receive. When using encapsulation over UDP there are at least two checksums that may be of interest: the encapsulated packet’s transport checksum, and the UDP checksum in the outer header.

18.1.2.1. Transmit checksum offload

NICs may provide a protocol agnostic method to offload transmit checksum (NETIF_F_HW_CSUM in Linux parlance) that can be used with UDP encapsulation. In this method the host provides checksum related parameters in a transmit descriptor for a packet. These parameters include the starting offset of data to checksum, the length of data to checksum, and the offset in the packet where the computed checksum is to be written. The host initializes the checksum field to pseudo header checksum. In the case of encapsulated packet, the checksum for an encapsulated transport layer packet, a TCP packet for instance, can be offloaded by setting the appropriate checksum parameters.

NICs typically can offload only one transmit checksum per packet, so simultaneously offloading both an inner transport packet’s checksum and the outer UDP checksum is likely not possible. In this case setting UDP checksum to zero (per above discussion) and offloading the inner transport packet checksum might be acceptable.

There is a proposal in [I-D.herbert-remotecsumoffload] to leverage NIC checksum offload when an encapsulator is co-resident with a host.

18.1.2.2. Receive checksum offload

Protocol encapsulation is compatible with NICs that perform a protocol agnostic receive checksum (CHECKSUM_COMPLETE in Linux parlance). In this technique, a NIC computes a ones complement checksum over all (or some predefined portion) of a packet. The

computed value is provided to the host stack in the packet's receive descriptor. The host driver can use this checksum to "patch up" and validate any inner packet transport checksum, as well as the outer UDP checksum if it is non-zero.

Many legacy NICs don't provide checksum-complete but instead provide an indication that a checksum has been verified (CHECKSUM_UNNECESSARY in Linux). Usually, such validation is only done for simple TCP/IP or UDP/IP packets. If a NIC indicates that a UDP checksum is valid, the checksum-complete value for the UDP packet is the "not" of the pseudo header checksum. In this way, checksum-unnecessary can be converted to checksum-complete. So if the NIC provides checksum-unnecessary for the outer UDP header in an encapsulation, checksum conversion can be done so that the checksum-complete value is derived and can be used by the stack to validate an checksums in the encapsulated packet.

18.1.3. Segmentation offload

Segmentation offload refers to techniques that attempt to reduce CPU utilization on hosts by having the transport layers of the stack operate on large packets. In transmit segmentation offload, a transport layer creates large packets greater than MTU size (Maximum Transmission Unit). It is only at much lower point in the stack, or possibly the NIC, that these large packets are broken up into MTU sized packet for transmission on the wire. Similarly, in receive segmentation offload, small packets are coalesced into large, greater than MTU size packets at a point low in the stack receive path or possibly in a device. The effect of segmentation offload is that the number of packets that need to be processed in various layers of the stack is reduced, and hence CPU utilization is reduced.

18.1.3.1. Transmit Segmentation Offload

Transmit Segmentation Offload (TSO) is a NIC feature where a host provides a large (larger than MTU size) TCP packet to the NIC, which in turn splits the packet into separate segments and transmits each one. This is useful to reduce CPU load on the host.

The process of TSO can be generalized as:

- o Split the TCP payload into segments which allow packets with size less than or equal to MTU.
- o For each created segment:

  1. Replicate the TCP header and all preceding headers of the original packet.

2. Set payload length fields in any headers to reflect the length of the segment.
3. Set TCP sequence number to correctly reflect the offset of the TCP data in the stream.
4. Recompute and set any checksums that either cover the payload of the packet or cover header which was changed by setting a payload length.

Following this general process, TSO can be extended to support TCP encapsulation UDP. For each segment the Ethernet, outer IP, UDP header, encapsulation header, inner IP header if tunneling, and TCP headers are replicated. Any packet length header fields need to be set properly (including the length in the outer UDP header), and checksums need to be set correctly (including the outer UDP checksum if being used).

To facilitate TSO with encapsulation it is recommended that optional fields should not contain values that must be updated on a per segment basis-- for example an encapsulation header should not include checksums, lengths, or sequence numbers that refer to the payload. If the encapsulation header does not contain such fields then the TSO engine only needs to copy the bits in the encapsulation header when creating each segment and does not need to parse the encapsulation header.

18.1.3.2. Large Receive Offload

Large Receive Offload (LRO) is a NIC feature where packets of a TCP connection are reassembled, or coalesced, in the NIC and delivered to the host as one large packet. This feature can reduce CPU utilization in the host.

LRO requires significant protocol awareness to be implemented correctly and is difficult to generalize. Packets in the same flow need to be unambiguously identified. In the presence of tunnels or network virtualization, this may require more than a five-tuple match (for instance packets for flows in two different virtual networks may have identical five-tuples). Additionally, a NIC needs to perform validation over packets that are being coalesced, and needs to fabricate a single meaningful header from all the coalesced packets.

The conservative approach to supporting LRO for encapsulation would be to assign packets to the same flow only if they have identical five-tuple and were encapsulated the same way. That is the outer IP addresses, the outer UDP ports, encapsulated protocol, encapsulation headers, and inner five tuple are all identical.

18.1.3.3. In summary:

In summary, for NIC offload:

- o The considerations for using full UDP checksums are different for NIC offload than for implementations in forwarding devices like routers and switches.
- o Be judicious about encapsulations that change fields on a per-packet basis, since such behavior might make it hard to use TSO.

19. Middlebox Considerations

This document has touched upon middleboxes in different section. The reason for this is as encapsulations get widely deployed one would expect different forms of middleboxes might become aware of the encapsulation protocol just as middleboxes have been made aware of other protocols where there are business and deployment opportunities. Such middleboxes are likely to do more than just drop packets based on the UDP port number used by an encapsulation protocol.

We note that various forms of encapsulation gateways that stitch one encapsulation protocol together with another form of protocol could have similar effects.

An example of a middlebox that could see some use would be an NVO3-aware firewall that would filter on the VNI IDs to provide some defense in depth inside or across NVO3 datacenters.

A question for the IETF is whether we should document what to do or what not to do in such middleboxes. This document touches on areas of OAM and ECMP as it relates to middleboxes and it might make sense to document how encapsulation-aware middleboxes should avoid unintended consequences in those (and perhaps other) areas.

In summary:

- o We are likely to see middleboxes that at least parse the headers for successful new encapsulations.
- o Should the IETF document considerations for what not to do in such middleboxes?

20. Related Work

The IETF and industry has defined encapsulations for a long time, with examples like GRE [RFC2890], VXLAN [RFC7348], and NVGRE [I-D.sridharan-virtualization-nvgre] being able to carry arbitrary Ethernet payloads, and various forms of IP-in-IP and IPsec

encapsulations that can carry IP packets. As part of NVO3 there has been additional proposals like Geneve [I-D.gross-geneve] and GUE [I-D.herbert-gue] which look at more extensibility. NSH [I-D.quinn-sfc-nsh] is an example of an encapsulation that tries to provide extensibility mechanisms which target both hardware and software implementations.

There is also a large body of work around MPLS encapsulations [RFC3032]. The MPLS-in-UDP work [I-D.ietf-mpls-in-udp] and GRE over UDP [I-D.ietf-tsvwg-gre-in-udp-encap] have worked on some of the common issues around checksum and congestion control. MPLS also introduced a entropy label [RFC6790]. There is also a proposal for MPLS encryption [I-D.farrelll-mpls-opportunistic-encrypt].

The idea to use a UDP encapsulation with a UDP source port for entropy for the underlay routers' ECMP dates back to LISP [RFC6830].

The pseudo-wire work [RFC3985] is interesting in the notion of layering additional services/characteristics such as ordered delivery or timely deliver on top of an encapsulation. That layering approach might be useful for the new encapsulations as well. For instance, the control word [RFC4385]. There is also material on congestion control for pseudo-wires in [I-D.ietf-pwe3-congcons].

Both MPLS and L2TP [RFC3931] rely on some control or signaling to establish state (for the path/labels in the case of MPLS, and for the session in the case of L2TP). The NVO3, SFC, and BIER encapsulations will also have some separation between the data plane and control plane, but the type of separation appears to be different.

IEEE 802.1 has defined encapsulations for L2 over L2, in the form of Provider backbone Bridging (PBB) [IEEE802.1Q-2014] and Equal Cost Multipath (ECMP) [IEEE802.1Q-2014]. The latter includes something very similar to the way the UDP source port is used as entropy: "The flow hash, carried in an F-TAG, serves to distinguish frames belonging to different flows and can be used in the forwarding process to distribute frames over equal cost paths"

TRILL, which is also a L2 over L2 encapsulation, took a different approach to entropy but preserved the ability for OAM frames [RFC7174] to use the same entropy hence ECMP path as data frames. In [I-D.ietf-trill-oam-fm] there 96 bytes of headers for entropy in the OAM frames, followed by the actual OAM content. This ensures that any headers, which fit in those 96 bytes except the OAM bit in the TRILL header, can be used for ECMP hashing.

As encapsulations evolve there might be a desire to fit multiple inner packets into one outer packet. The work in [I-D.saldana-tsvwg-simplemux] might be interesting for that purpose.

21. Acknowledgements

The authors acknowledge the comments from Alia Atlas, Fred Baker, David Black, Bob Briscoe, Stewart Bryant, Mike Cox, Andy Malis, Radia Perlman, Carlos Pignataro, Jamal Hadi Salim, Michael Smith, and Lucy Yong.

22. Open Issues

- o Middleboxes:
  - * Due to OAM there are constraints on middleboxes in general. If middleboxes inspect the packet past the outer IP+UDP and encapsulation header and look for inner IP and TCP/UDP headers, that might violate the assumption that OAM packets will be handled the same as regular data packets. That issue is broader than just QoS - applies to firewall filters etc.
  - * Firewalls looking at inner payload? How does that work for OAM frames? Even if it only drops ... TRILL approach might be an option? Would that encourage more middleboxes making the network more fragile?
  - * Editorially perhaps we should pull the above two into a separate section about middlebox considerations?
- o Next-protocol indication - should it be common across different encapsulation headers? We will have different ways to indicate the presence of the first encapsulation header in a packet (could be a UDP destination port, an Ethernet type, etc depending on the outer delivery header). But for the next protocol past an encapsulation header one could envision creating or adoption a common scheme. Such a would also need to be able to identify following headers like Ethernet, IPv4/IPv6, ESP, etc.
- o Common OAM error reporting protocol?
- o There is discussion about timestamps, sequence numbers, etc in three different parts of the document. OAM, Congestion Considerations, and Service Model, where the latter argues that a pseudo-wire service should really be layered on top of the encapsulation using its own header. Those recommendations seem to be at odds with each other. Do we envision sequence numbers, timestamps, etc as potential extensions for OAM and CC? If so, those extensions could be used to provide a service which doesn’t reorder packets.

23. Change Log

The changes from draft-rtg-dt-encap-01 based on feedback at the Dallas IETF meeting:

- o Setting the context that not all common issues might apply to all encapsulations, but that they should all be understood before being dismissed.
- o Clarified that IPv6 flow label is useful for entropy in combination with a UDP source port.
- o Editorially added a "summary" set of bullets to most sections.
- o Editorial clarifications in the next protocol section to more clearly state the three areas.
- o Folded the two next protocol sections into one.
- o Mention the MPLS first nibble issue in the next protocol section.
- o Mention that viewing the next protocol as an index to a table with processing instructions can provide additional flexibility in the protocol evolution.
- o For the OAM "don’t forward to end stations" added that defining a bit seems better than using a special next-protocol value.
- o Added mention of DTLS in addition to IPsec for security.
- o Added some mention of IPv6 hob-by-hop options of other headers than potentially can be copied from inner to outer header.
- o Added text on architectural considerations when it might make sense to define an additional header/protocol as opposed to using the extensibility mechanism in the existing encapsulation protocol.
- o Clarified the "unconstrained TLVs" in the hardware friendly section.
- o Clarified the text around checksum verification and full vs. header checksums.
- o Added wording that the considerations might apply for encaps outside of the routing area.
- o Added references to draft-ietf-pwe3-congcons, draft-ietf-tsvwg-rfc5405bis, RFC2473, and RFC7325
- o Removed reference to RFC3948.
- o Updated the acknowledgements section.
- o Added this change log section.

24. References

24.1. Normative References

[I-D.ietf-tsvwg-rfc5405bis]
Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", draft-ietf-tsvwg-rfc5405bis-19 (work in progress), October 2016.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <http://www.rfc-editor.org/info/rfc2473>.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, DOI 10.17487/RFC2890, September 2000, <http://www.rfc-editor.org/info/rfc2890>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <http://www.rfc-editor.org/info/rfc2983>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <http://www.rfc-editor.org/info/rfc3032>.

[RFC3931] Lau, J., Ed., Townsley, M., Ed., and I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, DOI 10.17487/RFC3931, March 2005, <http://www.rfc-editor.org/info/rfc3931>.

[RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <http://www.rfc-editor.org/info/rfc3985>.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <http://www.rfc-editor.org/info/rfc4385>.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <http://www.rfc-editor.org/info/rfc6040>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <http://www.rfc-editor.org/info/rfc6438>.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <http://www.rfc-editor.org/info/rfc6790>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <http://www.rfc-editor.org/info/rfc6830>.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <http://www.rfc-editor.org/info/rfc6935>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <http://www.rfc-editor.org/info/rfc6936>.

[RFC7174] Salam, S., Senevirathne, T., Aldrin, S., and D. Eastlake 3rd, "Transparent Interconnection of Lots of Links (TRILL) Operations, Administration, and Maintenance (OAM) Framework", RFC 7174, DOI 10.17487/RFC7174, May 2014, <http://www.rfc-editor.org/info/rfc7174>.

[RFC7325] Villamizar, C., Ed., Kompella, K., Amante, S., Malis, A., and C. Pignataro, "MPLS Forwarding Compliance and Performance Requirements", RFC 7325, DOI 10.17487/RFC7325, August 2014, <http://www.rfc-editor.org/info/rfc7325>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

[RFC7364] Narten, T., Ed., Gray, E., Ed., Black, D., Fang, L., Kreeger, L., and M. Napierala, "Problem Statement: Overlays for Network Virtualization", RFC 7364, DOI 10.17487/RFC7364, October 2014, <http://www.rfc-editor.org/info/rfc7364>.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, DOI 10.17487/RFC7365, October 2014, <http://www.rfc-editor.org/info/rfc7365>.

24.2. Informative References

[I-D.briscoe-conex-data-centre]
Briscoe, B. and M. Sridharan, "Network Performance Isolation in Data Centres using Congestion Policing", draft-briscoe-conex-data-centre-02 (work in progress), February 2014.

[I-D.farrelll-mpls-opportunistic-encrypt]
Farrel, A. and S. Farrell, "Opportunistic Security in MPLS Networks", draft-farrelll-mpls-opportunistic-encrypt-05 (work in progress), June 2015.

[I-D.gross-geneve]
Gross, J., Sridhar, T., Garg, P., Wright, C., Ganga, I., Agarwal, P., Duda, K., Dutt, D., and J. Hudson, "Geneve: Generic Network Virtualization Encapsulation", draft-gross-geneve-02 (work in progress), October 2014.

[I-D.herbert-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-herbert-gue-03 (work in progress), March 2015.

[I-D.herbert-remotecsumoffload]
Herbert, T., "Remote checksum offload for encapsulation", draft-herbert-remotecsumoffload-02 (work in progress), March 2016.

[I-D.ietf-mpls-in-udp]
Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", draft-ietf-mpls-in-udp-11 (work in progress), January 2015.

[I-D.ietf-nvo3-arch]
Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data Center Network Virtualization Overlays (NVO3)", draft-ietf-nvo3-arch-08 (work in progress), September 2016.

[I-D.ietf-pwe3-congcons]
Stein, Y., Black, D., and B. Briscoe, "Pseudowire Congestion Considerations", draft-ietf-pwe3-congcons-02 (work in progress), July 2014.

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-11 (work in progress), July 2015.

[I-D.ietf-sfc-problem-statement]
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-13 (work in progress), February 2015.

[I-D.ietf-trill-oam-fm]
Senevirathne, T., Finn, N., Salam, S., Kumar, D., Eastlake, D., Aldrin, S., and L. Yizhou, "TRILL Fault Management", draft-ietf-trill-oam-fm-11 (work in progress), October 2014.

[I-D.ietf-tsvwg-circuit-breaker]
Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-15 (work in progress), April 2016.

[I-D.ietf-tsvwg-gre-in-udp-encap]
Yong, L., Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", draft-ietf-tsvwg-gre-in-udp-encap-19 (work in progress), September 2016.

[I-D.ietf-tsvwg-port-use]
Touch, J., "Recommendations on Using Assigned Transport Port Numbers", draft-ietf-tsvwg-port-use-11 (work in progress), April 2015.

[I-D.quinn-sfc-nsh]
Quinn, P., Guichard, J., Surendra, S., Smith, M., Henderickx, W., Nadeau, T., Agarwal, P., Manur, R., Chauhan, A., Halpern, J., Majee, S., Elzur, U., Melman, D., Garg, P., McConnell, B., Wright, C., and K. Kevin, "Network Service Header", draft-quinn-sfc-nsh-07 (work in progress), February 2015.

[I-D.saldana-tsvwg-simplemux]
Saldana, J., "Simplemux. A generic multiplexing protocol", draft-saldana-tsvwg-simplemux-05 (work in progress), July 2016.

[I-D.shepherd-bier-problem-statement]
Shepherd, G., Dolganow, A., and a. arkadiy.gulko@thomsonreuters.com, "Bit Indexed Explicit Replication (BIER) Problem Statement", draft-shepherd-bier-problem-statement-02 (work in progress), February 2015.

[I-D.sridharan-virtualization-nvgre]
Garg, P. and Y. Wang, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-08 (work in progress), April 2015.

[I-D.wei-tsvwg-tunnel-congestion-feedback]
Wei, X., Zhu, L., Deng, L., and B. Briscoe, "Tunnel Congestion Feedback", draft-wei-tsvwg-tunnel-congestion-feedback-04 (work in progress), June 2015.

[I-D.wijnands-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-wijnands-bier-architecture-05 (work in progress), March 2015.

[I-D.wijnands-mpls-bier-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-wijnands-mpls-bier-encapsulation-02 (work in progress), December 2014.

[I-D.xu-bier-encapsulation]
Xu, X., somasundaram.s@alcatel-lucent.com, s., Jacquenet, C., Raszuk, R., and Z. Zhang, "A Transport-Independent Bit Index Explicit Replication (BIER) Encapsulation Header", draft-xu-bier-encapsulation-06 (work in progress), September 2016.

[IEEE802.1Q-2014]
IEEE, "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks", IEEE Std 802.1Q-2014, 2014, <http://www.ieee802.org/1/pages/802.1Q-2014.html>.

(Access Controlled link within page)

[RFC0033] Crocker, S., "New Host-Host Protocol", RFC 33, DOI 10.17487/RFC0033, February 1970, <http://www.rfc-editor.org/info/rfc33>.

Authors' Addresses

Erik Nordmark
Arista Networks
5453 Great America Parkway
Santa Clara, CA 95054
USA

Email: nordmark@arista.com

Albert Tian
Ericsson Inc.
300 Holger Way
San Jose, California 95134
USA

Email: albert.tian@ericsson.com

Jesse Gross
VMware
3401 Hillview Ave.
Palo Alto, CA 94304
USA

Email: jgross@vmware.com

Jon Hudson
Brocade Communications Systems, Inc.
130 Holger Way
San Jose, CA 95134
USA

Email: jon.hudson@gmail.com

Lawrence Kreeger
Cisco Systems, Inc.
170 W. Tasman Avenue
San Jose, CA 95134
USA

Email: kreeger@cisco.com

Pankaj Garg
Microsoft
1 Microsoft Way
Redmond, WA 98052
USA

Email: pankajg@microsoft.com

Patricia Thaler
Broadcom Corporation
3151 Zanker Road
San Jose, CA 95134
USA

Email: pthaler@broadcom.com

Tom Herbert
Facebook
1 Hacker Way
Menlo Park, CA 94052
USA

Email: tom@herbertland.com

TSVWG
Internet-Draft
Updates: 4787, 5382, 5508 (if approved)
Intended status: Best Current Practice
Expires: September 3, 2016

R. Penno
Cisco
S. Perreault
Jive Communications
M. Boucadair, Ed.
Orange
S. Sivakumar
Cisco
K. Naito
NTT
March 2, 2016

Network Address Translation (NAT) Behavioral Requirements Updates
draft-ietf-tsvwg-behave-requirements-update-08

Abstract

This document clarifies and updates several requirements of RFC4787, RFC5382, and RFC5508 based on operational and development experience. The focus of this document is NAT44.

This document updates RFCs 4787, 5382, and 5508.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[RFC4787], [RFC5382], and [RFC5508] contributed to enhance Network Address Translation (NAT) interoperability and conformance. Operational experience gained through widespread deployment and evolution of NAT indicates that some areas of the original documents need further clarification or updates. This document provides such clarifications and updates.

1.1. Scope

The goal of this document is to clarify and update the set of requirements listed in [RFC4787], [RFC5382], and [RFC5508]. The document focuses exclusively on NAT44.

The scope of this document has been set so that it does not create new requirements beyond those specified in the documents cited above.

Carrier-Grade NAT (CGN) related requirements are defined in [RFC6888].

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The reader is assumed to be familiar with the terminology defined in: [RFC2663],[RFC4787],[RFC5382], and [RFC5508].

In this document, the term "NAT" refers to both "Basic NAT" and "Network Address/Port Translator (NAPT)" (see Section 3 of [RFC4787]). As a reminder, Basic NAT and NAPT are two variations of traditional NAT, in that translation in Basic NAT is limited to IP addresses alone, whereas translation in NAPT is extended to include IP address and Transport identifier (such as TCP/UDP port or ICMP query ID) (refer to Section 2 of [RFC3022]).

2. TCP Session Tracking

[RFC5382] specifies TCP timers associated with various connection states but does not specify the TCP state machine a NAT44 should follow as a basis to apply such timers.

Update: The TCP state machine depicted in Figure 1, adapted from [RFC6146], SHOULD be implemented by a NAT for TCP session tracking purposes.

```
                   +----------------------------+
                   |                            |
                   V                            |
                +------+   Client               |
                |CLOSED|-----SYN------+         |
                +------+              |         |
                    ^                 |         |
                    |TCP_TRANS T.O.   |         |
                    |                 V         |
                +-------+          +-------+    |
                | TRANS |          |  INIT |    |
                +-------+          +-------+    |
                  |    ^               |        |
             data pkt   |               |        |
                  | Server/Client RST  |        |
                  |  TCP_EST T.O.      |        |
                  V    |           Server SYN   |
              +--------------+          |        |
              | ESTABLISHED  |<---------+        |
              +--------------+                   |
               |           |                     |
         Client FIN    Server FIN                |
               |           |                     |
               V           V                     |
        +---------+   +----------+               |
        |  C FIN  |   |  S FIN   |               |
        |   RCV   |   |    RCV   |               |
        +---------+   +----------+               |
            |             |                      |
        Server FIN      Client FIN            TCP_TRANS
            |             |                      T.O.
            V             V                       |
        +----------------------+                  |
        |   C FIN + S FIN RCV  |-----------------+
        +----------------------+
    Legend:
      * Messages sent or received from the server are
        prefixed with "Server".
      * Messages sent or received from the client are
        prefixed with "Client".
      * "C" means "Client-side"
      * "S" means "Server-side".
      * TCP_EST T.O: refers to the established connection
        idle timeout as defined in [RFC5382].
      * TCP_TRANS T.O: refers to the transitory connection
        idle timeout as defined in [RFC5382].
```

Figure 1: Simplified version of the TCP State Machine

2.1. TCP Transitory Connection Idle-Timeout

The transitory connection idle-timeout is defined as the minimum time a TCP connection in the partially open or closing phases must remain idle before the NAT considers the associated session a candidate for removal (REQ-5 of [RFC5382]). But [RFC5382] does not clearly state whether these can be configured separately.

Clarification: This document clarifies that a NAT SHOULD provide different configurable parameters for configuring the open and closing idle timeouts.

To accommodate deployments that consider a partially open timeout of 4 minutes as being excessive from a security standpoint, a NAT MAY allow the configured timeout to be less than 4 minutes. However, a minimum default transitory connection idle-timeout of 4 minutes is RECOMMENDED.

2.2. TCP RST

[RFC5382] leaves the handling of TCP RST packets unspecified.

Update: This document adopts a similar default behavior as in [RFC6146]. Concretely, when the NAT receives a TCP RST matching an existing mapping, it MUST translate the packet according to the NAT mapping entry. Moreover, the NAT SHOULD wait for 4 minutes before deleting the session and removing any state associated with it if no packets are received during that 4 minutes timeout.

Notes:

* Admittedly, the NAT has to verify whether received TCP RST packets belong to a connection. This verification check is required to avoid off-path attacks.

* If the NAT removes immediately the NAT mapping upon receipt of a TCP RST message, stale connections may be maintained by endpoints if the first RST message is lost between the NAT and the recipient.

3. Port Overlapping Behavior

REQ-1 from [RFC4787] and REQ-1 from [RFC5382] specify a specific port overlapping behavior; that is the external IP address and port can be reused for connections originating from the same internal source IP address and port irrespective of the destination. This is known as endpoint-independent mapping (EIM).

Update: This document clarifies that this port overlapping behavior may be extended to connections originating from different internal source IP addresses and ports as long as their destinations are different.

The following mechanism MAY be implemented by a NAT:

If destination addresses and ports are different for outgoing connections started by local clients, a NAT MAY assign the same external port as the source ports for the connections. The port overlapping mechanism manages mappings between external packets and internal packets by looking at and storing their 5-tuple (protocol, source address, source port, destination address, destination port).

This enables concurrent use of a single NAT external port for multiple transport sessions, which allows a NAT to successfully process packets in an IP address resource limited network (e.g., deployment with high address space multiplicative factor (refer to Appendix B. of [RFC6269])).

4. Address Pooling Paired (APP)

The "IP address pooling" behavior of "Paired" (APP) was recommended in REQ-2 from [RFC4787], but the behavior when an external IPv4 runs out of ports was left undefined.

Clarification: This document clarifies that if APP is enabled, new sessions from a host that already has a mapping associated with an external IP that ran out of ports SHOULD be dropped. A configuration parameter MAY be provided to allow a NAT to starting using ports from another external IP address when the one that anchored the APP mapping ran out of ports. Tweaking this configuration parameter is a trade-off between service continuity and APP strict enforcement. Note, this behavior is sometimes referred as ’soft-APP’.

As a reminder, the recommendation for the particular case of a CGN is that an implementation must use the same external IP address mapping for all sessions associated with the same internal IP address, be they TCP, UDP, ICMP, something else, or a mix of different protocols [RFC6888].

Update: This behavior SHOULD apply also for TCP.

5. Endpoint-Independent Mapping (EIM) Protocol Independence

REQ-1 from [RFC4787] and REQ-1 from [RFC5382] do not specify whether EIM are protocol-dependent or protocol-independent. For example, if an outbound TCP SYN creates a mapping, it is left undefined whether outbound UDP packets can reuse such mapping.

Update: EIM mappings SHOULD be protocol-dependent. A configuration parameter MAY be provided to allow protocols that multiplex TCP and UDP over the same source IP address and port number to use a single mapping. The default value of this configuration parameter MUST be protocol-dependent EIM.

This update is consistent with the stateful NAT64 [RFC6146] that clearly specifies three binding information bases (TCP, UDP, ICMP).

6. Endpoint-Independent Filtering (EIF) Protocol Independence

REQ-8 from [RFC4787] and REQ-3 from [RFC5382] do not specify whether mappings with endpoint-independent filtering (EIF) are protocol-independent or protocol-dependent. For example, if an outbound TCP SYN creates a mapping, it is left undefined whether inbound UDP packets matching that mapping should be accepted or rejected.

Update: EIF filtering SHOULD be protocol-dependent. A configuration parameter MAY be provided to make it protocol-independent. The default value of this configuration parameter MUST be protocol-dependent EIF.

This behavior is aligned with the update in Section 5.

Applications that can be transported over a variety of transport protocols and/or support transport fall back schemes won’t experience connectivity failures if the NAT is configured with protocol-independent EIM and protocol-independent EIF.

7. Endpoint-Independent Filtering (EIF) Mapping Refresh

The NAT mapping Refresh direction may have a "NAT Inbound refresh behavior" of "True" according to REQ-6 from [RFC4787], but [RFC4787] does not clarify how this behavior applies to EIF mappings. The issue in question is whether inbound packets that match an EIF mapping but do not create a new session due to a security policy should refresh the mapping timer.

Clarification: This document clarifies that even when a NAT has an inbound refresh behavior set to ’TRUE’, such packets SHOULD NOT

refresh the mapping. Otherwise a simple attack of a packet every 2 minutes can keep the mapping indefinitely.

Update: This behavior SHOULD apply also for TCP.

7.1. Outbound Mapping Refresh and Error Packets

Update: In the case of NAT outbound refresh behavior, ICMP Errors or TCP RST outbound packets, sent as response to inbound packets, SHOULD NOT refresh the mapping. Other packets which indicate the host is not interested in receiving packets MAY be configurable to also not refresh state, such as STUN error response [RFC5389] or IKE INVALID_SYNTAX [RFC7296].

8. Port Parity

Update: A NAT MAY disable port parity preservation for all dynamic mappings. Nevertheless, A NAT SHOULD support means to explicitly request to preserve port parity (e.g., [RFC7753]).

Note: According to [RFC6887], dynamic mappings are said to be dynamic in the sense that they are created on demand, either implicitly or explicitly:

1. Implicit dynamic mappings refer to mappings that are created as a side effect of traffic such as an outgoing TCP SYN or outgoing UDP packet. Implicit dynamic mappings usually have a finite lifetime, though this lifetime is generally not known to the client using them.

2. Explicit dynamic mappings refer to mappings that are created as a result, for example, of explicit Port Control Protocol (PCP) MAP and PEER requests. Explicit dynamic mappings have a finite lifetime, and this lifetime is communicated to the client.

9. Port Randomization

Update: A NAT SHOULD follow the recommendations specified in Section 4 of [RFC6056], especially:

> "A NAPT that does not implement port preservation [RFC4787] [RFC5382] SHOULD obfuscate selection of the ephemeral port of a packet when it is changed during translation of that packet. A NAPT that does implement port preservation SHOULD obfuscate the ephemeral port of a packet only if the port must be changed as a result of the port being already in use for some other session. A NAPT that performs parity preservation and that

must change the ephemeral port during translation of a packet SHOULD obfuscate the ephemeral ports. The algorithms described in this document could be easily adapted such that the parity is preserved (i.e., force the lowest order bit of the resulting port number to 0 or 1 according to whether even or odd parity is desired)."

10. IP Identification (IP ID)

Update: A NAT SHOULD handle the Identification field of translated IPv4 packets as specified in Section 5.3.1 of [RFC6864].

11. ICMP Query Mappings Timeout

Section 3.1 of [RFC5508] specifies that ICMP Query Mappings are to be maintained by a NAT. However, the specification doesn’t discuss Query Mapping timeout values. Section 3.2 of [RFC5508] only discusses ICMP Query Session Timeouts.

Update: ICMP Query Mappings MAY be deleted once the last session using the mapping is deleted.

12. Hairpinning Support for ICMP Packets

REQ-7 from [RFC5508] specifies that a NAT enforcing ’Basic NAT’ must support traversal of hairpinned ICMP Query sessions.

Clarification: This implicitly means that address mappings from external address to internal address (similar to Endpoint Independent Filters) must be maintained to allow inbound ICMP Query sessions. If an ICMP Query is received on an external address, a NAT can then translate to an internal IP.

REQ-7 from [RFC5508] specifies that all NATs must support the traversal of hairpinned ICMP Error messages.

Clarification: This behavior requires a NAT to maintain address mappings from external IP address to internal IP address in addition to the ICMP Query Mappings described in Section 3.1 of [RFC5508].

13. IANA Considerations

This document does not require any IANA action.

14. Security Considerations

NAT behavioral considerations are discussed in [RFC4787], [RFC5382], and [RFC5508].

Because some of the clarifications and updates (e.g., Section 2) are inspired from NAT64, the security considerations discussed in Section 5 of [RFC6146] apply also for this specification.

The update in Section 3 allows for an optimized NAT resource usage. In order to avoid service disruption, the NAT must not invoke this functionality unless the packets are to be sent to distinct destination addresses.

Some of the updates (e.g., Section 7, Section 9, and Section 11) allow for an increased security compared to [RFC4787], [RFC5382], and [RFC5508]. Particularly:

- The updates in Section 7 and Section 11 prevent an illegitimate node to maintain mappings activated in the NAT while these mappings should be cleared.

- Port randomization (Section 9) complicates tracking hosts located behind a NAT.

Section 4 and Section 12 propose updates that increase the serviceability of a host located behind a NAT. These updates do not introduce any additional security concerns to [RFC4787], [RFC5382], and [RFC5508].

The updates in Section 5 and Section 6 allow for a better NAT transparency from an application standpoint. Hosts that require a restricted filtering behavior should enable specific policies (e.g., access control list (ACL)) either locally or by soliciting a dedicated security device (e.g., firewall). How a host updates its filtering policies is out of scope of this document.

The update in Section 8 induces security concerns that are specific to the protocol used to interact with the NAT. For example, if PCP is used to explicitly request parity preservation for a given mapping, the security considerations discussed in [RFC6887] should be taken into account.

The update in Section 10 may have undesired effects on the performance of the NAT in environments in which fragmentation is massively experienced. Such issue may be used as an attack vector against NATs.

15. References

15.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <http://www.rfc-editor.org/info/rfc4787>.

[RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008, <http://www.rfc-editor.org/info/rfc5382>.

[RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, DOI 10.17487/RFC5508, April 2009, <http://www.rfc-editor.org/info/rfc5508>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <http://www.rfc-editor.org/info/rfc6056>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <http://www.rfc-editor.org/info/rfc6146>.

[RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field", RFC 6864, DOI 10.17487/RFC6864, February 2013, <http://www.rfc-editor.org/info/rfc6864>.

15.2. Informative References

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, DOI 10.17487/RFC2663, August 1999, <http://www.rfc-editor.org/info/rfc2663>.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <http://www.rfc-editor.org/info/rfc3022>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

[RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <http://www.rfc-editor.org/info/rfc6269>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <http://www.rfc-editor.org/info/rfc6887>.

[RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888, April 2013, <http://www.rfc-editor.org/info/rfc6888>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

[RFC7753] Sun, Q., Boucadair, M., Sivakumar, S., Zhou, C., Tsou, T., and S. Perreault, "Port Control Protocol (PCP) Extension for Port-Set Allocation", RFC 7753, DOI 10.17487/RFC7753, February 2016, <http://www.rfc-editor.org/info/rfc7753>.

Acknowledgements

Thanks to Dan Wing, Suresh Kumar, Mayuresh Bakshi, Rajesh Mohan, Lars Eggert, Gorry Fairhurst, Brandon Williams, and David Black for their review and discussion.

Many thanks to Ben Laurie for the secdir review, and Dan Romascanu for the Gen-ART review.

Dan Wing proposed some text for the configurable errors in Section 7.1.

Contributors

The following individual contributed text to the document:

Sarat Kamiset, Insieme Networks, United States

Authors' Addresses

Reinaldo Penno
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: repenno@cisco.com

Simon Perreault
Jive Communications
Canada

Email: sperreault@jive.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Senthil Sivakumar
Cisco Systems, Inc.
United States

Email: ssenthil@cisco.com

Kengo Naito
NTT
Tokyo
Japan

Email: k.naito@nttv6.jp

TSVWG Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: October 6, 2016

G. Fairhurst
University of Aberdeen

April 04, 2016

Network Transport Circuit Breakers
draft-ietf-tsvwg-circuit-breaker-15

Abstract

This document explains what is meant by the term "network transport Circuit Breaker" (CB). It describes the need for circuit breakers for network tunnels and applications when using non-congestion-controlled traffic, and explains where circuit breakers are, and are not, needed. It also defines requirements for building a circuit breaker and the expected outcomes of using a circuit breaker within the Internet.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The term "Circuit Breaker" originates in electricity supply, and has nothing to do with network circuits or virtual circuits. In electricity supply, a Circuit Breaker is intended as a protection mechanism of last resort. Under normal circumstances, a Circuit Breaker ought not to be triggered; it is designed to protect the supply network and attached equipment when there is overload. People do not expect an electrical circuit-breaker (or fuse) in their home to be triggered, except when there is a wiring fault or a problem with an electrical appliance.

In networking, the Circuit Breaker (CB) principle can be used as a protection mechanism of last resort to avoid persistent excessive congestion impacting other flows that share network capacity. Persistent congestion was a feature of the early Internet of the 1980s. This resulted in excess traffic starving other connections from access to the Internet. It was countered by the requirement to use congestion control (CC) in the Transmission Control Protocol (TCP) [Jacobsen88]. These mechanisms operate in Internet hosts to cause TCP connections to "back off" during congestion. The addition of a congestion control to TCP (currently documented in [RFC5681] ensured the stability of the Internet, because it was able to detect congestion and promptly react. This was effective in an Internet where most TCP flows were long-lived (ensuring that they could detect and respond to congestion before the flows terminated). Although TCP was by far the dominant traffic, this is no longer the always the case, and non-congestion-controlled traffic, including many applications using the User Datagram Protocol (UDP), can form a significant proportion of the total traffic traversing a link. The current Internet therefore requires that non-congestion-controlled traffic is considered to avoid persistent excessive congestion.

A network transport Circuit Breaker is an automatic mechanism that is used to continuously monitor a flow or aggregate set of flows. The mechanism seeks to detect when the flow(s) experience persistent excessive congestion. When this is detected, a Circuit Breaker terminates (or significantly reduce the rate of) the flow(s). This is a safety measure to prevent starvation of network resources denying other flows from access to the Internet. Such measures are essential for an Internet that is heterogeneous and for traffic that is hard to predict in advance. Avoiding persistent excessive congestion is important to reduce the potential for "Congestion Collapse" [RFC2914].

There are important differences between a transport Circuit Breaker and a congestion control method. Congestion control (as implemented in TCP, SCTP, and DCCP) operates on a timescale on the order of a packet round-trip-time (RTT), the time from sender to destination and return. Congestion at a network link can also be detected using Explicit Congestion Notification (ECN) [RFC3168], which allows the network to signal congestion by marking ECN-capable packets with a Congestion Experienced (CE) mark. Both loss and reception of CE-marked packets are treated as congestion events. Congestion control methods are able to react to a congestion event by continuously adapting to reduce their transmission rate. The goal is usually to limit the transmission rate to a maximum rate that reflects a fair use of the available capacity across a network path. These methods typically operate on individual traffic flows (e.g., a 5-tuple that includes the IP addresses, protocol, and ports).

In contrast, Circuit Breakers are recommended for non-congestion-controlled Internet flows and for traffic aggregates, e.g., traffic sent using a network tunnel. They operate on timescales much longer than the packet RTT, and trigger under situations of abnormal (excessive) congestion. People have been implementing what this document characterizes as circuit breakers on an ad hoc basis to protect Internet traffic. This document therefore provides guidance on how to deploy and use these mechanisms. Later sections provide examples of cases where circuit-breakers may or may not be desirable.

A Circuit Breaker needs to measure (meter) some portion of the traffic to determine if the network is experiencing congestion and needs to be designed to trigger robustly when there is persistent excessive congestion.

A Circuit Breaker trigger will often utilize a series of successive sample measurements metered at an ingress point and an egress point (either of which could be a transport endpoint). The trigger needs to operate on a timescale much longer than the path round trip time (e.g., seconds to possibly many tens of seconds). This longer period is needed to provide sufficient time for transport congestion control (or applications) to adjust their rate following congestion, and for the network load to stabilize after any adjustment. Congestion events can be common when a congestion-controlled transport is used over a network link operating near capacity. Each event results in reduction in the rate of the transport flow experiencing congestion. The longer period seeks to ensure that a Circuit Breaker does not accidentally trigger following a single (or even successive) congestion events.

Once triggered, the Circuit Breaker needs to provide a control function (called the "reaction"). This removes traffic from the network, either by disabling the flow or by significantly reducing the level of traffic. This reaction provides the required protection to prevent persistent excessive congestion being experienced by other flows that share the congested part of the network path.

Section 4 defines requirements for building a Circuit Breaker.

The operational conditions that cause a Circuit Breaker to trigger ought to be regarded as abnormal. Examples of situations that could trigger a Circuit Breaker include:

- o anomalous traffic that exceeds the provisioned capacity (or whose traffic characteristics exceed the threshold configured for the Circuit Breaker);

- o traffic generated by an application at a time when the provisioned network capacity is being utilised for other purposes;

- o routing changes that cause additional traffic to start using the path monitored by the Circuit Breaker;

- o misconfiguration of a service/network device where the capacity available is insufficient to support the current traffic aggregate;

- o misconfiguration of an admission controller or traffic policer that allows more traffic than expected across the path monitored by the Circuit Breaker.

Other mechanisms could also be available to network operators to detect excessive congestion (e.g., an observation of excessive utilisation for a port on a network device). Utilising such information, operational mechanisms could react to reduce network load over a shorter timescale than those of a network transport Circuit Breaker. The role of the Circuit Breaker over such paths remains as a method of last resort. Because it acts over a longer timescale, the Circuit Breaker ought to trigger only when other reactions did not succeed in reducing persistent excessive congestion.

In many cases, the reason for triggering a Circuit Breaker will not be evident to the source of the traffic (user, application, endpoint, etc). A Circuit Breaker can be used to limit traffic from applications that are unable, or choose not, to use congestion control, or where the congestion control properties of the traffic cannot be relied upon (e.g., traffic carried over a network tunnel). In such circumstances, it is all but impossible for the Circuit Breaker to signal back to the impacted applications. In some cases applications could therefore have difficulty in determining that a Circuit Breaker has triggered, and where in the network this happened.

Application developers are therefore advised, where possible, to deploy appropriate congestion control mechanisms. An application that uses congestion control will be aware of congestion events in the network. This allows it to regulate the network load under congestion, and is expected to avoid triggering a network Circuit Breaker. For applications that can generate elastic traffic, this will often be a preferred solution.

1.1. Types of Circuit Breaker

There are various forms of network transport circuit breaker. These are differentiated mainly on the timescale over which they are triggered, but also in the intended protection they offer:

- o Fast-Trip Circuit Breakers: The relatively short timescale used by this form of circuit breaker is intended to provide protection for network traffic from a single flow or related group of flows.

- o Slow-Trip Circuit Breakers: This circuit breaker utilizes a longer timescale and is designed to protect network traffic from congestion by traffic aggregates.

- o Managed Circuit Breakers: Utilize the operations and management functions that might be present in a managed service to implement a circuit breaker.

Examples of each type of circuit breaker are provided in section 4.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Design of a Circuit-Breaker (What makes a good circuit breaker?)

Although circuit breakers have been talked about in the IETF for many years, there has not yet been guidance on the cases where circuit breakers are needed or upon the design of circuit breaker mechanisms. This document seeks to offer advice on these two topics.

Circuit Breakers are RECOMMENDED for IETF protocols and tunnels that carry non-congestion-controlled Internet flows and for traffic aggregates. This includes traffic sent using a network tunnel. Designers of other protocols and tunnel encapsulations also ought to consider the use of these techniques as a last resort to protect traffic that shares the network path being used.

This document defines the requirements for design of a Circuit Breaker and provides examples of how a Circuit Breaker can be constructed. The specifications of individual protocols and tunnel encapsulations need to detail the protocol mechanisms needed to implement a Circuit Breaker.

Section 3.1 describes the functional components of a circuit breaker and section 3.2 defines requirements for implementing a Circuit Breaker.

3.1. Functional Components

The basic design of a Circuit Breaker involves communication between an ingress point (a sender) and an egress point (a receiver) of a network flow or set of flows. A simple picture of operation is provided in figure 1. This shows a set of routers (each labelled R) connecting a set of endpoints.

A Circuit Breaker is used to control traffic passing through a subset of these routers, acting between the ingress and a egress point network devices. The path between the ingress and egress could be provided by a tunnel or other network-layer technique. One expected use would be at the ingress and egress of a service, where all traffic being considered terminates beyond the egress point, and hence the ingress and egress carry the same set of flows.

```
 +--------+                                                  +--------+
 |Endpoint|                                                  |Endpoint|
 +--+-----+          >>> circuit breaker traffic >>>         +--+-----+
    |                                                           |
    | +-+  +-+  +---------+  +-+  +-+  +-+  +--------+  +-+  +-+ |
    +-+R+--+R+->+ Ingress +--+R+--+R+--+R+--+ Egress |--+R+--+R+-+
      +++  +-+  +------+--+  +-+  +-+  +-+  +-----+--+  +++  +-+
       |         ^     |                         |      |
       |         |  +--+------+            +------+--+   |
       |         |  | Ingress |            | Egress  |   |
       |         |  | Meter   |            | Meter   |   |
       |         |  +----+----+            +----+----+   |
       |         |       |                      |        |
  +-+  |         |  +----+----+                 |        |  +-+
  |R+--+         |  | Measure +<----------------+        +--+R|
  +++            |  +----+----+      Reported               +++
   |             |       |           Egress                  |
   |             |  +----+----+      Measurement             |
+--+-----+       |  | Trigger +                          +--+-----+
|Endpoint|       |  +----+----+                          |Endpoint|
+--------+       |       |                               +--------+
                 +---<---+
                  Reaction
```

Figure 1: A CB controlling the part of the end-to-end path between an ingress point and an egress point. (Note: In some cases, the trigger

and measurement functions could alternatively be located at other locations (e.g., at a network operations centre.)

In the context of a Circuit Breaker, the ingress and egress functions could be implemented in different places. For example, they could be located in network devices at a tunnel ingress and at the tunnel egress. In some cases, they could be located at one or both network endpoints (see figure 2), implemented as components within a transport protocol.

```
 +----------+                 +----------+
 | Ingress  |  +-+  +-+  +-+  | Egress   |
 | Endpoint +->+R+--+R+--+R+--+ Endpoint |
 +--+----+--+  +-+  +-+  +-+  +----+-----+
    ^    |                          |
    | +--+------+             +----+----+
    | | Ingress |             | Egress  |
    | | Meter   |             | Meter   |
    | +----+----+             +----+----+
    |      |                          |
    | +--- +----+                     |
    | | Measure +<-----------------+
    | +----+----+      Reported
    |      |           Egress
    | +----+----+      Measurement
    | | Trigger |
    | +----+----+
    |      |
    +---<--+
    Reaction
```

Figure 2: An endpoint CB implemented at the sender (ingress) and receiver (egress).

The set of components needed to implement a Circuit Breaker are:

1. An ingress meter (at the sender or tunnel ingress) that records the number of packets/bytes sent in each measurement interval. This measures the offered network load for a flow or set of flows. For example, the measurement interval could be many seconds (or every few tens of seconds or a series of successive shorter measurements that are combined by the Circuit Breaker Measurement function).

2. An egress meter (at the receiver or tunnel egress) that records the number/bytes received in each measurement interval. This measures the supported load for the flow or set of flows, and

could utilize other signals to detect the effect of congestion (e.g., loss/congestion marking [RFC3168] experienced over the path). The measurements at the egress could be synchronised (including an offset for the time of flight of the data, or referencing the measurements to a particular packet) to ensure any counters refer to the same span of packets.

3. A method that communicates the measured values at the ingress and egress to the Circuit Breaker Measurement function. This could use several methods including: Sending return measurement packets (or control messages) from a receiver to a trigger function at the sender; an implementation using Operations, Administration and Management (OAM); or sending an in-band signalling datagram to the trigger function. This could also be implemented purely as a control plane function, e.g., using a software-defined network controller.

4. A measurement function that combines the ingress and egress measurements to assess the present level of network congestion. (For example, the loss rate for each measurement interval could be deduced from calculating the difference between ingress and egress counter values.) Note the method does not require high accuracy for the period of the measurement interval (or therefore the measured value, since isolated and/or infrequent loss events need to be disregarded.)

5. A trigger function that determines whether the measurements indicate persistent excessive congestion. This function defines an appropriate threshold for determining that there is persistent excessive congestion between the ingress and egress. This preferably considers a rate or ratio, rather than an absolute value (e.g., more than 10% loss, but other methods could also be based on the rate of transmission as well as the loss rate). The Circuit Breaker is triggered when the threshold is exceeded in multiple measurement intervals (e.g., 3 successive measurements). Designs need to be robust so that single or spurious events do not trigger a reaction.

6. A reaction that is applied at the Ingress when the Circuit Breaker is triggered. This seeks to automatically remove the traffic causing persistent excessive congestion.

7. A feedback control mechanism that triggers when either the receive or ingress and egress measurements are not available, since this also could indicate a loss of control packets (also a symptom of heavy congestion or inability to control the load).

3.2. Other network topologies

A Circuit Breaker can be deployed in networks with topologies different to that presented in figures 1 and 2. This section describes examples of such usage, and possible places where functions can be implemented.

3.2.1. Use with a multicast control/routing protocol

```
 +----------+                 +--------+  +----------+
 | Ingress  |  +-+  +-+  +-+  | Egress |  |  Egress  |
 | Endpoint +->+R+--+R+--+R+--+ Router |--+ Endpoint +->+
 +----+-----+  +-+  +-+  +-+  +---+--+-+  +----+-----+  |
      ^         ^    ^    ^       |  ^         |        |
      |         |    |    |       |  |         |        |
 +----+----+    + - - - < - - - - + |    +----+----+   | Reported
 | Ingress |      multicast Prune  |    | Egress  |   | Ingress
 | Meter   |                       |    | Meter   |   | Measurement
 +---------+                       |    +----+----+   |
                                   |         |        |
                                   |    +----+----+   |
                                   |    | Measure +<--+
                                   |    +----+----+
                                   |         |
                                   |    +----+----+
                             multicast |    | Trigger |
                             Leave     |    +----+----+
                             Message   |         |
                                      +----<----+
```

Figure 3: An example of a multicast CB controlling the end-to-end path between an ingress endpoint and an egress endpoint.

Figure 3 shows one example of how a multicast Circuit Breaker could be implemented at a pair of multicast endpoints (e.g., to implement a Fast-Trip Circuit Breaker, Section 5.1). The ingress endpoint (the sender that sources the multicast traffic) meters the ingress load, generating an ingress measurement (e.g., recording timestamped packet counts), and sends this measurement to the multicast group together with the traffic it has measured.

Routers along a multicast path forward the multicast traffic (including the ingress measurement) to all active endpoint receivers. Each last hop (egress) router forwards the traffic to one or more egress endpoint(s).

In this figure, each endpoint includes a meter that performs a local egress load measurement. An endpoint also extracts the received ingress measurement from the traffic, and compares the ingress and egress measurements to determine if the Circuit Breaker ought to be triggered. This measurement has to be robust to loss (see previous section). If the Circuit Breaker is triggered, it generates a multicast leave message for the egress (e.g., an IGMP or MLD message sent to the last hop router), which causes the upstream router to cease forwarding traffic to the egress endpoint [RFC1112].

Any multicast router that has no active receivers for a particular multicast group will prune traffic for that group, sending a prune message to its upstream router. This starts the process of releasing the capacity used by the traffic and is a standard multicast routing function (e.g., using Protocol Independent Multicast Sparse Mode (PIM-SM) routing protocol [RFC4601]). Each egress operates autonomously, and the Circuit Breaker "reaction" is executed by the multicast control plane (e.g., by PIM) requiring no explicit signalling by the Circuit Breaker along the communication path used for the control messages. Note: there is no direct communication with the Ingress, and hence a triggered Circuit Breaker only controls traffic downstream of the first hop multicast router. It does not stop traffic flowing from the sender to the first hop router; this is common practice for multicast deployment.

The method could also be used with a multicast tunnel or subnetwork (e.g., Section 5.2, Section 5.3), where a meter at the ingress generates additional control messages to carry the measurement data towards the egress where the egress metering is implemented.

3.2.2. Use with control protocols supporting pre-provisioned capacity

Some paths are provisioned using a control protocol, e.g., flows provisioned using the Multi-Protocol Label Switching (MPLS) services, paths provisioned using the resource reservation protocol (RSVP), networks utilizing Software Defined Network (SDN) functions, or admission-controlled Differentiated Services. Figure 1 shows one expected use case, where in this usage a separate device could be used to perform the measurement and trigger functions. The reaction generated by the trigger could take the form of a network control message sent to the ingress and/or other network elements causing these elements to react to the Circuit Breaker. Examples of this type of use are provided in section Section 5.3.

3.2.3. Unidirectional Circuit Breakers over Controlled Paths

A Circuit Breaker can be used to control uni-directional UDP traffic, providing that there is a communication path that can be used for control messages to connect the functional components at the Ingress and Egress. This communication path for the control messages can exist in networks for which the traffic flow is purely unidirectional. For example, a multicast stream that sends packets across an Internet path and can use multicast routing to prune flows to shed network load. Some other types of subnetwork also utilize control protocols that can be used to control traffic flows.

4. Requirements for a Network Transport Circuit Breaker

The requirements for implementing a Circuit Breaker are:

1. There needs to be a communication path for control messages to carry measurement data from the ingress meter and from the egress meter to the point of measurement. (Requirements 16-18 relate to the transmission of control messages.)

2. A CB is REQUIRED to define a measurement period over which the CB Measurement function measures the level of congestion or loss. This method does not have to detect individual packet loss, but MUST have a way to know that packets have been lost/ marked from the traffic flow.

3. An egress meter can also count ECN [RFC3168] congestion marks as a part of measurement of congestion, but in this case, loss MUST also be measured to provide a complete view of the level of congestion. For tunnels, [ID-ietf-tsvwg-tunnel-congestion-feedback] describes a way to measure both loss and ECN-marking; these measurements could be used on a relatively short timescale to drive a congestion control response and/or aggregated over a longer timescale with a higher trigger threshold to drive a CB. Subsequent bullet items in this section discuss the necessity of using a longer timescale and a higher trigger threshold.

4. The measurement period used by a CB Measurement function MUST be longer than the time that current Congestion Control algorithms need to reduce their rate following detection of congestion. This is important because end-to-end Congestion Control algorithms require at least one RTT to notify and adjust the traffic when congestion is experienced, and congestion bottlenecks can share traffic with a diverse range of RTTs. The measurement period is therefore expected to be significantly longer than the RTT experienced by the CB itself.

5. If necessary, a CB MAY combine successive individual meter samples from the ingress and egress to ensure observation of an average measurement over a sufficiently long interval. (Note when meter samples need to be combined, the combination needs to reflect the sum of the individual sample counts divided by the total time/volume over which the samples were measured. Individual samples over different intervals can not be directly combined to generate an average value.)

6. A CB MUST be constructed so that it does not trigger under light or intermittent congestion (see requirements 7-9).

7. A CB is REQUIRED to define a threshold to determine whether the measured congestion is considered excessive.

8. A CB is REQUIRED to define the triggering interval, defining the period over which the trigger uses the collected measurements. CBs need to trigger over a sufficiently long period to avoid additionally penalizing flows with a long path RTT (e.g., many path RTTs).

9. A CB MUST be robust to multiple congestion events. This usually will define a number of measured persistent congestion events per triggering period. For example, a CB MAY combine the results of several measurement periods to determine if the CB is triggered (e.g., it is triggered when persistent excessive congestion is detected in 3 of the measurements within the triggering interval).

10. The normal reaction to a trigger SHOULD disable all traffic that contributed to congestion (otherwise, see requirements 11,12).

11. The reaction MUST be much more severe than that of a Congestion Control algorithm (such as TCP’s congestion control [RFC5681] or TCP-Friendly Rate Control, TFRC [RFC5348]), because the CB reacts to more persistent congestion and operates over longer timescales (i.e., the overload condition will have persisted for a longer time before the CB is triggered).

12. A reaction that results in a reduction SHOULD result in reducing the traffic by at least an order of magnitude. A response that achieves the reduction by terminating flows, rather than randomly dropping packets, will often be more desirable to users of the service. A CB that reduces the rate of a flow, MUST continue to monitor the level of congestion and MUST further react to reduce the rate if the CB is again triggered.

13. The reaction to a triggered CB MUST continue for a period that is at least the triggering interval. Operator intervention will usually be required to restore a flow. If an automated response is needed to reset the trigger, then this needs to not be immediate. The design of an automated reset mechanism needs to be sufficiently conservative that it does not adversely interact with other mechanisms (including other CB algorithms that control traffic over a common path). It SHOULD NOT perform an automated reset when there is evidence of continued congestion.

14. A CB trigger SHOULD be regarded as an abnormal network event. As such, this event SHOULD be logged. The measurements that lead to triggering of the CB SHOULD also be logged.

15. The control communication needs to carry measurements (requirement 1) and, in some uses, also needs to transmit trigger messages to the ingress. This control communication may be in-band or out-of-band. The use of in-band communication is RECOMMENDED when either design would be possible. The preferred CB design is one that triggers when it fails to receive measurement reports that indicate an absence of congestion, in contrast to relying on the successful transmission of a "congested" signal back to the sender. (The feedback signal could itself be lost under congestion).

    in-Band: An in-band control method SHOULD assume that loss of control messages is an indication of potential congestion on the path, and repeated loss ought to cause the CB to be triggered. This design has the advantage that it provides fate-sharing of the traffic flow(s) and the control communications. This fate-sharing property is weaker when some or all of the measured traffic is sent using a path that differs from the path taken by the control traffic (e.g., where traffic and control messages follow a different path due to use of equal-cost multipath routing, traffic engineering, or tunnels for specific types of traffic).

    Out-of-Band: An out-of-band control method SHOULD NOT trigger CB reaction when there is loss of control messages (e.g., a loss of measurements). This avoids failure amplification/ propagation when the measurement and data paths fail independently. A failure of an out-of-band communication path SHOULD be regarded as abnormal network event and be handled as appropriate for the network, e.g., this event SHOULD be logged, and additional network operator action might be appropriate, depending on the network and the traffic involved.

16. The control communication MUST be designed to be robust to packet loss. A control message can be lost if there is a failure of the communication path used for the control messages, loss is likely to also be experienced during congestion/ overload. This does not imply that it is desirable to provide reliable delivery (e.g., over TCP), since this can incur additional delay in responding to congestion. Appropriate mechanisms could be to duplicate control messages to provide increased robustness to loss, or/and to regard a lack of control traffic as an indication that excessive congestion could be being experienced [ID-ietf-tsvwg-RFC5405.bis]. If control messages traffic are sent over a shared path, it is RECOMMENDED that this control traffic is prioritized to reduce the probability of loss under congestion. Control traffic also needs to be considered when provisioning a network that uses a Circuit Breaker.

17. There are security requirements for the control communication between endpoints and/or network devices (Section 7). The authenticity of the source and integrity of the control messages (measurements and triggers) MUST be protected from off-path attacks. When there is a risk of on-path attack, a cryptographic authentication mechanism for all control/ measurement messages is RECOMMENDED.

5. Examples of Circuit Breakers

There are multiple types of Circuit Breaker that could be defined for use in different deployment cases. There could be cases where a flow become controlled by multiple Circuit Breakers (e.g., when the traffic of an end-to-end flow is carried in a tunnel within the network). This section provides examples of different types of Circuit Breaker:

5.1. A Fast-Trip Circuit Breaker

[RFC2309] discusses the dangers of congestion-unresponsive flows and states that "all UDP-based streaming applications should incorporate effective congestion avoidance mechanisms". Some applications do not use a full-featured transport (TCP, SCTP, DCCP). These applications (e.g., using UDP and its UDP-Lite variant) need to provide appropriate congestion avoidance. Guidance for applications that do not use congestion-controlled transports is provided in [ID-ietf-tsvwg-RFC5405.bis]. Such mechanisms can be designed to react on much shorter timescales than a Circuit Breaker, that only observes a traffic envelope. Congestion control methods can also interact with an application to more effectively control its sending rate.

A fast-trip Circuit Breaker is the most responsive form of Circuit Breaker. It has a response time that is only slightly larger than that of the traffic that it controls. It is suited to traffic with well-understood characteristics (and could include one or more trigger functions specifically tailored the type of traffic for which it is designed). It is not suited to arbitrary network traffic and could be unsuitable for traffic aggregates, since it could prematurely trigger (e.g., when the combined traffic from multiple congestion-controlled flows leads to short-term overload).

Although the mechanisms can be implemented in RTP-aware network devices, these mechanisms are also suitable for implementation in endpoints (e.g., as a part of the transport system) where they can also compliment end-to-end congestion control methods. A shorter response time enables these mechanisms to triggers before other forms of Circuit Breaker (e.g., Circuit Breakers operating on traffic aggregates at a point along the network path).

5.1.1. A Fast-Trip Circuit Breaker for RTP

A set of fast-trip Circuit Breaker methods have been specified for use together by a Real-time Transport Protocol (RTP) flow using the RTP/AVP Profile [RTP-CB]. It is expected that, in the absence of severe congestion, all RTP applications running on best-effort IP networks will be able to run without triggering these Circuit Breakers. A fast-trip RTP Circuit Breaker is therefore implemented as a fail-safe that when triggered will terminate RTP traffic.

The sending endpoint monitors reception of in-band RTP Control Protocol (RTCP) reception report blocks, as contained in SR or RR packets, that convey reception quality feedback information. This is used to measure (congestion) loss, possibly in combination with ECN [RFC6679].

The Circuit Breaker action (shutdown of the flow) is triggered when any of the following trigger conditions are true:

1. An RTP Circuit Breaker triggers on reported lack of progress.

2. An RTP Circuit Breaker triggers when no receiver reports messages are received.

3. An RTP Circuit Breaker triggers when the long-term RTP throughput (over many RTTs) exceeds a hard upper limit determined by a method that resembles TCP-Friendly Rate Control (TFRC).

4. An RTP Circuit Breaker includes the notion of Media Usability. This Circuit Breaker is triggered when the quality of the

transported media falls below some required minimum acceptable quality.

5.2. A Slow-trip Circuit Breaker

A slow-trip Circuit Breaker could be implemented in an endpoint or network device. This type of Circuit Breaker is much slower at responding to congestion than a fast-trip Circuit Breaker. This is expected to be more common.

One example where a slow-trip Circuit Breaker is needed is where flows or traffic-aggregates use a tunnel or encapsulation and the flows within the tunnel do not all support TCP-style congestion control (e.g., TCP, SCTP, TFRC), see [ID-ietf-tsvwg-RFC5405.bis] section 3.1.3. A use case is where tunnels are deployed in the general Internet (rather than "controlled environments" within an Internet service provider or enterprise network), especially when the tunnel could need to cross a customer access router.

5.3. A Managed Circuit Breaker

A managed Circuit Breaker is implemented in the signalling protocol or management plane that relates to the traffic aggregate being controlled. This type of Circuit Breaker is typically applicable when the deployment is within a "controlled environment".

A Circuit Breaker requires more than the ability to determine that a network path is forwarding data, or to measure the rate of a path - which are often normal network operational functions. There is an additional need to determine a metric for congestion on the path and to trigger a reaction when a threshold is crossed that indicates persistent excessive congestion.

The control messages can use either in-band or out-of-band communications.

5.3.1. A Managed Circuit Breaker for SAToP Pseudo-Wires

[RFC4553], SAToP Pseudo-Wires (PWE3), section 8 describes an example of a managed Circuit Breaker for isochronous flows.

If such flows were to run over a pre-provisioned (e.g., Multi-Protocol Label Switching, MPLS) infrastructure, then it could be expected that the Pseudowire (PW) would not experience congestion, because a flow is not expected to either increase (or decrease) their rate. If, instead, PW traffic is multiplexed with other traffic over the general Internet, it could experience congestion. [RFC4553] states: "If SAToP PWs run over a PSN providing best-effort service,

they SHOULD monitor packet loss in order to detect "severe congestion". The currently recommended measurement period is 1 second, and the trigger operates when there are more than three measured Severely Errored Seconds (SES) within a period. If such a condition is detected, a SAToP PW ought to shut down bidirectionally for some period of time...".

The concept was that when the packet loss ratio (congestion) level increased above a threshold, the PW was by default disabled. This use case considered fixed-rate transmission, where the PW had no reasonable way to shed load.

The trigger needs to be set at the rate that the PW was likely to experience a serious problem, possibly making the service non-compliant. At this point, triggering the Circuit Breaker would remove the traffic preventing undue impact on congestion-responsive traffic (e.g., TCP). Part of the rationale, was that high loss ratios typically indicated that something was "broken" and ought to have already resulted in operator intervention, and therefore need to trigger this intervention.

An operator-based response to triggering of a Circuit Breaker provides an opportunity for other action to restore the service quality, e.g., by shedding other loads or assigning additional capacity, or to consciously avoid reacting to the trigger while engineering a solution to the problem. This could require the trigger function to send a control message to a third location (e.g., a network operations centre, NOC) that is responsible for operation of the tunnel ingress, rather than the tunnel ingress itself.

5.3.2. A Managed Circuit Breaker for Pseudowires (PWs)

Pseudowires (PWs) [RFC3985] have become a common mechanism for tunneling traffic, and could compete for network resources both with other PWs and with non-PW traffic, such as TCP/IP flows.

[ID-ietf-pals-congcons] discusses congestion conditions that can arise when PWs compete with elastic (i.e., congestion responsive) network traffic (e.g, TCP traffic). Elastic PWs carrying IP traffic (see [RFC4488]) do not raise major concerns because all of the traffic involved responds, reducing the transmission rate when network congestion is detected.

In contrast, inelastic PWs (e.g., a fixed bandwidth Time Division Multiplex, TDM) [RFC4553] [RFC5086] [RFC5087]) have the potential to harm congestion responsive traffic or to contribute to excessive congestion because inelastic PWs do not adjust their transmission rate in response to congestion. [ID-ietf-pals-congcons] analyses TDM

PWs, with an initial conclusion that a TDM PW operating with a degree of loss that could result in congestion-related problems is also operating with a degree of loss that results in an unacceptable TDM service. For that reason, the document suggests that a managed Circuit Breaker that shuts down a PW when it persistently fails to deliver acceptable TDM service is a useful means for addressing these congestion concerns. (See Appendix A of [ID-ietf-pals-congcons] for further discussion.)

6. Examples where circuit breakers may not be needed.

A Circuit Breaker is not required for a single congestion-controlled flow using TCP, SCTP, TFRC, etc. In these cases, the congestion control methods are already designed to prevent persistent excessive congestion.

6.1. CBs over pre-provisioned Capacity

One common question is whether a Circuit Breaker is needed when a tunnel is deployed in a private network with pre-provisioned capacity.

In this case, compliant traffic that does not exceed the provisioned capacity ought not to result in persistent congestion. A Circuit Breaker will hence only be triggered when there is non-compliant traffic. It could be argued that this event ought never to happen - but it could also be argued that the Circuit Breaker equally ought never to be triggered. If a Circuit Breaker were to be implemented, it will provide an appropriate response if persistent congestion occurs in an operational network.

Implementing a Circuit Breaker will not reduce the performance of the flows, but in the event that persistent excessive congestion occurs it protects network traffic that shares network capacity with these flows. It also protects network traffic from a failure when Circuit Breaker traffic is (re)routed to cause additional network load on a non-pre-provisioned path.

6.2. CBs with tunnels carrying Congestion-Controlled Traffic

IP-based traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. A question therefore arises when people deploy a tunnel that is thought to only carry an aggregate of TCP traffic (or traffic using some other congestion control method): Is there advantage in this case in using a Circuit Breaker?

TCP (and SCTP) traffic in a tunnel is expected to reduce the transmission rate when network congestion is detected. Other transports (e.g, using UDP) can employ mechanisms that are sufficient to address congestion on the path [ID-ietf-tsvwg-RFC5405.bis]. However, even if the individual flows sharing a tunnel each implement a congestion control mechanism, and individually reduce their transmission rate when network congestion is detected, the overall traffic resulting from the aggregate of the flows does not necessarily avoid persistent congestion. For instance, most congestion control mechanisms require long-lived flows to react to reduce the rate of a flow. An aggregate of many short flows could result in many flows terminating before they experience congestion. It is also often impossible for a tunnel service provider to know that the tunnel only contains congestion-controlled traffic (e.g., Inspecting packet headers might not be possible). Some IP-based applications might not implement adequate mechanisms to address congestion. The important thing to note is that if the aggregate of the traffic does not result in persistent excessive congestion (impacting other flows), then the Circuit Breaker will not trigger. This is the expected case in this context - so implementing a Circuit Breaker ought not to reduce performance of the tunnel, but in the event that persistent excessive congestion occurs the Circuit Breaker protects other network traffic that shares capacity with the tunnel traffic.

6.3. CBs with Uni-directional Traffic and no Control Path

A one-way forwarding path could have no associated communication path for sending control messages, and therefore cannot be controlled using a Circuit Breaker (compare with Section 3.2.3).

A one-way service could be provided using a path with dedicated pre-provisioned capacity that is not shared with other elastic Internet flows (i.e., flows that vary their rate). A forwarding path could also be shared with other flows. One way to mitigate the impact of traffic on the other flows is to manage the traffic envelope by using ingress policing. Supporting this type of traffic in the general Internet requires operator monitoring to detect and respond to persistent excessive congestion.

7. Security Considerations

All Circuit Breaker mechanisms rely upon coordination between the ingress and egress meters and communication with the trigger function. This is usually achieved by passing network control information (or protocol messages) across the network. Timely operation of a Circuit Breaker depends on the choice of measurement period. If the receiver has an interval that is overly long, then

the responsiveness of the Circuit Breaker decreases. This impacts the ability of the Circuit Breaker to detect and react to congestion. If the interval is too short the Circuit Breaker could trigger prematurely resulting in insufficient time for other mechanisms to act, potentially resulting in unnecessary disruption to the service.

A Circuit Breaker could potentially be exploited by an attacker to mount a Denial of Service (DoS) attack against the traffic being controlled by the Circuit Breaker. Mechanisms therefore need to be implemented to prevent attacks on the network control information that would result in DoS.

The authenticity of the source and integrity of the control messages (measurements and triggers) MUST be protected from off-path attacks. Without protection, it could be trivial for an attacker to inject fake or modified control/measurement messages (e.g., indicating high packet loss rates) causing a Circuit Breaker to trigger and to therefore mount a DoS attack that disrupts a flow.

Simple protection can be provided by using a randomized source port, or equivalent field in the packet header (such as the RTP SSRC value and the RTP sequence number) expected not to be known to an off-path attacker. Stronger protection can be achieved using a secure authentication protocol to mitigate this concern.

An attack on the control messages is relatively easy for an attacker on the control path when the messages are neither encrypted nor authenticated. Use of a cryptographic authentication mechanism for all control/measurement messages is RECOMMENDED to mitigate this concern, and would also provide protection from off-path attacks. There is a design trade-off between the cost of introducing cryptographic security for control messages and the desire to protect control communication. For some deployment scenarios the value of additional protection from DoS attack will therefore lead to a requirement to authenticate all control messages.

Transmission of network control messages consumes network capacity. This control traffic needs to be considered in the design of a Circuit Breaker and could potentially add to network congestion. If this traffic is sent over a shared path, it is RECOMMENDED that this control traffic is prioritized to reduce the probability of loss under congestion. Control traffic also needs to be considered when provisioning a network that uses a Circuit Breaker.

The Circuit Breaker MUST be designed to be robust to packet loss that can also be experienced during congestion/overload. Loss of control messages could be a side-effect of a congested network, but also could arise from other causes Section 4.

The security implications depend on the design of the mechanisms, the type of traffic being controlled and the intended deployment scenario. Each design of a Circuit Breaker MUST therefore evaluate whether the particular Circuit Breaker mechanism has new security implications.

8. IANA Considerations

This document makes no request from IANA.

9. Acknowledgments

There are many people who have discussed and described the issues that have motivated this document. Contributions and comments included: Lars Eggert, Colin Perkins, David Black, Matt Mathis, Andrew McGregor, Bob Briscoe and Eliot Lear. This work was part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700).

10. Revision Notes

XXX RFC-Editor: Please remove this section prior to publication XXX

Draft 00

This was the first revision. Help and comments are greatly appreciated.

Draft 01

Contained clarifications and changes in response to received comments, plus addition of diagram and definitions. Comments are welcome.

WG Draft 00

Approved as a WG work item on 28th Aug 2014.

WG Draft 01

Incorporates feedback after Dallas IETF TSVWG meeting. This version is thought ready for WGLC comments. Definitions of abbreviations.

WG Draft 02

Minor fixes for typos. Rewritten security considerations section.

WG Draft 03

Updates following WGLC comments (see TSV mailing list). Comments from C Perkins; D Black and off-list feedback.

A clear recommendation of intended scope.

Changes include: Improvement of language on timescales and minimum measurement period; clearer articulation of endpoint and multicast examples - with new diagrams; separation of the controlled network case; updated text on position of trigger function; corrections to RTP-CB text; clarification of loss v ECN metrics; checks against submission checklist 9use of keywords, added meters to diagrams).

WG Draft 04

Added section on PW CB for TDM - a newly adopted draft (D. Black).

WG Draft 05

Added clarifications requested during AD review.

WG Draft 06

Fixed some remaining typos.

Update following detailed review by Bob Briscoe, and comments by D. Black.

WG Draft 07

Additional update following review by Bob Briscoe.

WG Draft 08

Updated text on the response to lack of meter measurements with managed circuit breakers. Additional comments from Eliot Lear (APPs area).

WG Draft 09

Updated text on applications from Eliot Lear. Additional feedback from Bob Briscoe.

WG Draft 10

Updated text following comments by D Black including a rewritten ECN requirements bullet with of a reference to a tunnel measurement method in [ID-ietf-tsvwg-tunnel-congestion-feedback].

WG Draft 11

Minor corrections after second WGLC.

WG Draft 12

Update following Gen-ART, RTG, and OPS review comments.

WG Draft 13

Fixed a typo.

WG Draft 14

Update after IESG discussion, including:

Reworded introduction. Added definition of ECN.

Requirement

Addressed inconsistency between requirements for control messages. - Removed a "MUST" - following WG feedback on a anearlier version of the draft that "SHOULD" is more appropriate.

Addressed comment about grouping requirements to help show they were inter-related. This reordered some requirements.

Reworded the security considerations.

Corrections to wording to improve clarity.

WG Draft 15 (incorporating pending corrections)

Corrected /applications might be implement/applications might not implement/

Corrected /Inspecting packet headers could/Inspecting packet headers might/

Removed Requirement 9, now duplicated (and renumbered remaining items).

Added "(See Appendix A of [ID-ietf-pals-congcons] for further discussion.)" to end of 5.3.2 - missed comment.

Simplified a sentence in section 6.1, without intended change of meaning.

Added a linking sentence to the second para of Section 6.3.

11. References

11.1. Normative References

[ID-ietf-tsvwg-RFC5405.bis]
Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines (Work-in-Progress)", 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

11.2. Informative References

[ID-ietf-pals-congcons]
Stein, YJ., Black, D., and B. Briscoe, "Pseudowire Congestion Considerations (Work-in-Progress)", 2015.

[ID-ietf-tsvwg-tunnel-congestion-feedback]
Wei, X., Zhu, L., and L. Dend, "Tunnel Congestion Feedback (Work-in-Progress)", 2015.

[Jacobsen88]
European Telecommunication Standards, Institute (ETSI), "Congestion Avoidance and Control", SIGCOMM Symposium proceedings on Communications architectures and protocols", August 1998.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <http://www.rfc-editor.org/info/rfc1112>.

[RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, DOI 10.17487/RFC2309, April 1998, <http://www.rfc-editor.org/info/rfc2309>.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <http://www.rfc-editor.org/info/rfc2914>.

[RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <http://www.rfc-editor.org/info/rfc3985>.

[RFC4488] Levin, O., "Suppression of Session Initiation Protocol (SIP) REFER Method Implicit Subscription", RFC 4488, DOI 10.17487/RFC4488, May 2006, <http://www.rfc-editor.org/info/rfc4488>.

[RFC4553] Vainshtein, A., Ed. and YJ. Stein, Ed., "Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP)", RFC 4553, DOI 10.17487/RFC4553, June 2006, <http://www.rfc-editor.org/info/rfc4553>.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <http://www.rfc-editor.org/info/rfc4601>.

[RFC5086] Vainshtein, A., Ed., Sasson, I., Metz, E., Frost, T., and P. Pate, "Structure-Aware Time Division Multiplexed (TDM) Circuit Emulation Service over Packet Switched Network (CESoPSN)", RFC 5086, DOI 10.17487/RFC5086, December 2007, <http://www.rfc-editor.org/info/rfc5086>.

[RFC5087] Stein, Y(J)., Shashoua, R., Insler, R., and M. Anavi, "Time Division Multiplexing over IP (TDMoIP)", RFC 5087, DOI 10.17487/RFC5087, December 2007, <http://www.rfc-editor.org/info/rfc5087>.

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <http://www.rfc-editor.org/info/rfc5348>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <http://www.rfc-editor.org/info/rfc6679>.

[RTP-CB] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions (draft-ietf-avtcore-rtp-circuit-breakers)", February 2014.

Author's Address

Godred Fairhurst
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen, Scotland AB24 3UE
UK

Email: gorry@erg.abdn.ac.uk
URI: http://www.erg.abdn.ac.uk

TSVWG
Internet-Draft
Intended status: Informational
Expires: June 18, 2017

R. Geib, Ed.
Deutsche Telekom
D. Black
Dell EMC

December 15, 2016

Diffserv-Interconnection classes and practice
draft-ietf-tsvwg-diffserv-intercon-14

Abstract

This document defines a limited common set of Diffserv Per Hop Behaviours (PHBs) and codepoints (DSCPs) to be applied at (inter)connections of two separately administered and operated networks, and explains how this approach can simplify network configuration and operation. Many network providers operate Multi Protocol Label Switching (MPLS) using Treatment Aggregates for traffic marked with different Diffserv Per Hop Behaviors, and use MPLS for interconnection with other networks. This document offers a simple interconnection approach that may simplify operation of Diffserv for network interconnection among providers that use MPLS and apply the Short-Pipe tunnel mode. While motivated by the requirements of MPLS network operators that use Short-Pipe tunnels, this document is applicable to other networks, both MPLS and non-MPLS.

Copyright Notice

Table of Contents

1. Introduction

Diffserv has been deployed in many networks; it provides differentiated traffic forwarding based on the Diffserv Codepoint (DSCP) field, which is part of the IP header [RFC2474]. This document defines a set of common Diffserv classes (Per Hop Behaviors, PHBs) and code points for use at interconnection points to which and from which locally used classes and code points should be mapped.

As described by section 2.3.4.2 of RFC2475, remarking of packets at domain boundaries is a Diffserv feature [RFC2475]. If traffic marked with unknown or unexpected DSCPs is received, RFC2474 recommends forwarding that traffic with default (best effort) treatment without changing the DSCP markings to better support incremental Diffserv deployment in existing networks as well as with routers that do not support Diffserv or are not configured to support it. Many networks do not follow this recommendation, and instead remark unknown or unexpected DSCPs to zero upon receipt for default (best effort) forwarding in accordance with the guidance in RFC2475 [RFC2475] to ensure that appropriate DSCPs are used within a Diffserv domain. This draft is based on the latter approach, and defines additional DSCPs that are known and expected at network interconnection interfaces in order to reduce the amount of traffic whose DSCPs are remarked to zero.

This document is motivated by requirements for IP network interconnection with Diffserv support among providers that operate Multi Protocol Label Switching (MPLS) in their backbones, but is also applicable to other technologies. The operational simplifications and methods in this document help align IP Diffserv functionality with MPLS limitations resulting from the widely deployed Short Pipe tunnel model for operation [RFC3270]. Further, limiting Diffserv to a small number of Treatment Aggregates can enable network traffic to leave a network with the DSCP value with which it was received, even if a different DSCP is used within the network, thus providing an opportunity to extend consistent Diffserv treatment across network boundaries.

In isolation, use of a defined set of interconnection PHBs and DSCPs may appear to be additional effort for a network operator. The primary offsetting benefit is that mapping from or to the interconnection PHBs and DSCPs is specified once for all of the interconnections to other networks that can use this approach. Absent this approach, the PHBs and DSCPs have to be negotiated and configured independently for each network interconnection, which has poor administrative and operational scaling properties. Further, consistent end-to-end Diffserv treatment is more likely to result when an interconnection code point scheme is used because traffic is remarked to the same PHBs at all network interconnections.

The interconnection approach described in this document (referred to as Diffserv-Intercon) uses a set of PHBs (mapped to four corresponding MPLS treatment aggregates) along with a set of interconnection DSCPs allowing straightforward rewriting to domain-internal DSCPs and defined DSCP markings for traffic forwarded to interconnected domains. The solution described here can be used in

other contexts benefitting from a defined Diffserv interconnection interface.

The basic idea is that traffic sent with a Diffserv-Interconnect PHB and DSCP is restored to that PHB and DSCP at each network interconnection, even though a different PHB and DSCP may be used by each network involved. The key requirement is that the network ingress interconnect DSCP be restored at network egress, and a key observation is that this is only feasible in general for a small number of DSCPs. Traffic sent with other DSCPs can be remarked to an interconnect DSCP or dealt with via additional agreement(s) among the operators of the interconnected networks; use of the MPLS Short Pipe model favors remarking unexpected DSCPs to zero in the absence of additional agreement(s), as explained further in this document.

In addition to the common interconnecting PHBs and DSCPs, interconnecting operators need to further agree on the tunneling technology used for interconnection (e.g., MPLS, if used) and control or mitigate the impacts of tunneling on reliability and MTU.

1.1. Related work

In addition to the activities that triggered this work, there are additional RFCs and Internet-drafts that may benefit from an interconnection PHB and DSCP scheme. RFC5160 suggests Meta-QoS-Classes to help enabling deployment of standardized end to end QoS classes [RFC5160]. The Diffserv-Intercon class- and codepoint scheme is intended to complement that work (e.g., by enabling a defined set of interconnection DSCPs and PHBs).

Border Gateway Protocol (BGP) signaling Class of Service at interconnection interfaces by BGP [I-D.knoll-idr-cos-interconnect], [ID.ietf-idr-sla] is complementary to Diffserv-Intercon. These two BGP documents focus on exchanging Service Level Agreement (SLA) and traffic conditioning parameters and assume that common PHBs identified by the signaled DSCPs have been established (e.g., via use of the Diffserv-Intercon DSCPs) prior to BGP signaling of PHB id codes.

1.2. Applicability Statement

This document is applicable to use of Differentiated Services for interconnection traffic between networks, and is particularly suited to interconnection of MPLS-based networks that use MPLS Short-pipe tunnels. This document is also applicable to other network technologies, but it is not intended for use within an individual network, where the approach specified in RFC5127 [RFC5127] is among the possible alternatives; see Section 3 for further discussion.

The Diffserv-Intercon approach described in this document simplifies IP based interconnection to domains operating the MPLS Short Pipe model for IP traffic, both terminating within the domain and transiting onward to another domain. Transiting traffic is received and sent with the same PHB and DSCP. Terminating traffic maintains the PHB with which it was received, however the DSCP may change.

Diffserv-Intercon is also applicable to the Pipe tunneling model [RFC2983], [RFC3270], but it is not applicable to the Uniform tunneling model [RFC2983], [RFC3270].

The Diffserv-Intercon approach defines a set of four PHBs for support at interconnections (or network boundaries in general). Corresponding DSCPs for use at an interconnection interface are added. Diffserv-intercon allows for a simple mapping of PHBs and DSCPs to MPLS Treatment Aggregates. It is extensible by IETF standardisation and this allows additional PHBs and DSCPs to be specified for the Diffserv-intercon scheme. Coding space for private interconnection agreements or provider internal services is left too.

1.3. Document Organization

This document is organized as follows: section 2 reviews the MPLS Short Pipe tunnel model for Diffserv Tunnels [RFC3270], because effective support for that model is a crucial goal of Diffserv-Intercon. Section 3 provides background on RFC5127’s approach to traffic class aggregation within a Diffserv network domain and contrasts it with the Diffserv-Intercon approach. Section 4 introduces Diffserv-Interconnection Treatment Aggregates, along with the PHBs and DSCPs that they use, and explains how other PHBs (and associated DSCPs) may be mapped to these Treatment Aggregates. Section 4 also discusses treatment of IP traffic, MPLS VPN Diffserv considerations and handling of high-priority Network Management traffic. Appendix A describes how the MPLS Short Pipe model (penultimate hop popping) impacts DSCP marking for IP interconnections.

2. MPLS and the Short Pipe tunnel model

This section provides a summary of the implications of the MPLS Short Pipe tunnels, and in particular their use of Penultimate Hop Popping (PHP, see RFC3270) on the Diffserv tunnel framework described in RFC2983. The Pipe and Uniform models for Differentiated Services and Tunnels are defined in [RFC2983]. RFC3270 adds the Short Pipe model to reflect the impact of MPLS PHP, primarily for MPLS-based IP tunnels and VPNs. The Short Pipe model and PHP have subsequently become popular with network providers that operate MPLS networks and

are now widely used to transport unencapsulated IP traffic. This has important implications for Diffserv functionality in MPLS networks.

RFC2474's recommendation to forward traffic with unrecognized DSCPs with Default (best effort) service without rewriting the DSCP has not been widely deployed in practice. Network operation and management are simplified when there is a 1-1 match between the DSCP marked on the packet and the forwarding treatment (PHB) applied by network nodes. When this is done, CS0 (the all-zero DSCP) is the only DSCP used for Default forwarding of best effort traffic, and a common practice is to remark to CS0 any traffic received with unrecognized or unsupported DSCPs at network edges.

MPLS networks are more subtle in this regard, as it is possible to encode the provider's DSCP in the MPLS Traffic Class (TC) field and allow that to differ from the PHB indicated by the DSCP in the MPLS-encapsulated IP packet. If the MPLS label with the provider's TC field is present at all hops within the provider network, this approach would allow an unrecognized DSCP to be carried edge-to-edge over an MPLS network, because the effective DSCP used by the provider's MPLS network would be encoded in the MPLS label TC field (and also carried edge-to-edge). Unfortunately this is only true for the Pipe tunnel model.

The Short Pipe tunnel model and PHP behave differently because PHP removes and discards the MPLS provider label carrying the provider's TC field before the traffic exits the provider's network. That discard occurs one hop upstream of the MPLS tunnel endpoint (which is usually at the network edge), resulting in no provider TC info being available at tunnel egress. To ensure consistent handling of traffic at the tunnel egress, the DSCP field in the MPLS-encapsulated IP header has to contain a DSCP that is valid for the provider's network, so that IP header cannot be used to carry a different DSCP edge-to-edge. See Appendix A for a more detailed discussion.

3. Relationship to RFC5127

This document draws heavily upon RFC5127's approach to aggregation of Diffserv traffic classes for use within a network, but there are important differences caused by characteristics of network interconnects that differ from links within a network.

3.1. RFC5127 Background

Many providers operate MPLS-based backbones that employ backbone traffic engineering to ensure that if a major link, switch, or router fails, the result will be a routed network that continues to function. Based on that foundation, [RFC5127] introduced the concept

of Diffserv Treatment Aggregates, which enable traffic marked with multiple DSCPs to be forwarded in a single MPLS Traffic Class (TC) based on robust provider backbone traffic engineering. This enables differentiated forwarding behaviors within a domain in a fashion that does not consume a large number of MPLS Traffic Classes.

RFC5127 provides an example aggregation of Diffserv service classes into 4 Treatment Aggregates. A small number of aggregates are used because:

- o The available coding space for carrying Traffic Class information (e.g., Diffserv PHB) in MPLS (and Ethernet) is only 3 bits in size, and is intended for more than just Diffserv purposes (see, e.g., [RFC5129]).

- o The common interconnection DSCPs ought not to use all 8 possible values. This leaves space for future standards, for private bilateral agreements and for local use PHBs and DSCPs.

- o Migrations from one Diffserv code point scheme to a different one is another possible application of otherwise unused DSCPs.

3.2. Differences from RFC5127

Like RFC5127, this document also uses four traffic aggregates, but differs from RFC5127 in some important ways:

- o It follows RFC2475 in allowing the DSCPs used within a network to differ from those to exchange traffic with other networks (at network edges), but provides support to restore ingress DSCP values if one of the recommended interconnect DSCPs in this draft is used. This results in DSCP remarking at both network ingress and network egress, and this draft assumes that such remarking at network edges is possible for all interface types.

- o Diffserv-Intercon suggests limiting the number of interconnection PHBs per Treatment Aggregate to the minimum required. As further discussed below, the number of PHBs per Treatment Aggregate is no more than two. When two PHBs are specified for a Diffserv-Intercon treatment aggregate, the expectation is that the provider network supports DSCPs for both PHBs, but uses a single MPLS TC for the Treatment Aggregate that contains the two PHBs.

- o Diffserv-Intercon suggests mapping other PHBs and DSCPs into the interconnection Treatment Aggregates as further discussed below.

- o Diffserv-Intercon treats network control traffic as a special case. Within a provider’s network, the CS6 DSCP is used for local

network control traffic (routing protocols and Operations, Administration, and Maintenance (OAM) traffic that is essential to network operation administration, control and management) that may be destined for any node within the network. In contrast, network control traffic exchanged between networks (e.g., BGP) usually terminates at or close to a network edge, and is not forwarded through the network because it is not part of internal routing or OAM for the receiving network. In addition, such traffic is unlikely to be covered by standard interconnection agreements; rather, it is more likely to be specifically configured (e.g., most networks impose restrictions on use of BGP with other networks for obvious reasons). See Section 4.2 for further discussion.

- o Because RFC5127 used a Treatment Aggregate for network control traffic, Diffserv-Intercon can instead define a fourth traffic aggregate to be defined for use at network interconnections instead of the Network Control aggregate in RFC5127. Network Control traffic may still be exchanged across network interconnections as further discussed in Section 4.2. Diffserv-Intercon uses this fourth traffic aggregate for VoIP traffic, where network-provided service differentiation is crucial, as even minor glitches are immediately apparent to the humans involved in the conversation.

4. The Diffserv-Intercon Interconnection Classes

At an interconnection, the networks involved need to agree on the PHBs used for interconnection and the specific DSCP for each PHB. This document defines a set of 4 interconnection Treatment Aggregates with well-defined DSCPs to be aggregated by them. A sending party remarks DSCPs from internal schemes to the interconnection code points. The receiving party remarks DSCPs to their internal scheme. The interconnect SLA defines the set of DSCPs and PHBs supported across the two interconnected domains and the treatment of PHBs and DSCPs that are not recognized by the receiving domain.

Similar approaches that use a small number of traffic aggregates (including recognition of the importance of VoIP traffic) have been taken in related standards and recommendations from outside the IETF, e.g., Y.1566 [Y.1566], GSMA IR.34 [IR.34]and MEF23.1 [MEF23.1].

The list of the four Diffserv-Interconnect traffic aggregates follows, highlighting differences from RFC5127 and suggesting mappings for all RFC4594 traffic classes to Diffserv-Intercon Treatment Aggregates:

Telephony Service Treatment Aggregate: PHB EF, DSCP 101 110 and PHB VOICE-ADMIT, DSCP 101 100, see [RFC3246], [RFC4594] and [RFC5865]. This Treatment Aggregate corresponds to RFC5127's real time Treatment Aggregate definition regarding the queuing (both delay and jitter should be minimized), but this aggregate is restricted to transport Telephony Service Class traffic in the sense of RFC4594 [RFC4594].

Bulk Real-Time Treatment Aggregate: This Treatment Aggregate is designed to transport PHB AF41, DSCP 100 010 (the other AF4 PHB group PHBs and DSCPs may be used for future extension of the set of DSCPs carried by this Treatment Aggregate). This Treatment Aggregate is intended for Diffserv-Intercon network interconnection of the portions of RFC5127's Real Time Treatment Aggregate, that consume significant bandwidth. This traffic is expected to consist of the RFC4594 classes Broadcast Video, Real-Time Interactive and Multimedia Conferencing. This treatment aggregate should be configured with a rate queue (consistent with RFC4594's recommendation for the transported traffic classes). By comparison to RFC5127, the number of DSCPs has been reduced to one (initially). The AF42 and AF43 PHBs could be added if there is a need for three-color marked Multimedia Conferencing traffic.

Assured Elastic Treatment Aggregate This Treatment Aggregate consists of PHBs AF31 and AF32 (i.e., DSCPs 011 010 and 011 100). By comparison to RFC5127, the number of DSCPs has been reduced to two. This document suggests to transport signaling marked by AF31 (e.g., as recommended by GSMA IR.34 [IR.34]). AF33 is reserved for extension of PHBs to be aggregated by this TA. For Diffserv-Intercon network interconnection, the following RFC4594 service classes should be mapped to the Assured Elastic Treatment Aggregate: the Signaling Service Class (being marked for lowest loss probability), Multimedia Streaming Service Class, the Low-Latency Data Service Class and the High-Throughput Data Service Class.

Default / Elastic Treatment Aggregate: transports the default PHB, CS0 with DSCP 000 000. RFC5127 example refers to this Treatment Aggregate as Aggregate Elastic. An important difference from RFC5127 is that any traffic with unrecognized or unsupported DSCPs may be remarked to this DSCP. For Diffserv-Intercon network interconnection, the RFC4594 standard service class and Low-priority Data service class should be mapped to this Treatment Aggregate. This document does not specify an interconnection class for RFC4594 Low-

priority data. This data may be forwarded by a Lower Effort PHB in one domain (like the PHB proposed by Informational [RFC3662]), but using the methods specified in this document will be remarked with DSCP CS0 at a Diffserv-Intercon network interconnection. This has the effect that Low-priority data is treated the same as data sent using the default class. (Note: In a network that implements RFC2474, Low-priority traffic marked as CS1 would otherwise receive better treatment than traffic using the default class.)

RFC2475 states that Ingress nodes must condition all inbound traffic to ensure that the DS codepoints are acceptable; packets found to have unacceptable codepoints must either be discarded or must have their DS codepoints modified to acceptable values before being forwarded. For example, an ingress node receiving traffic from a domain with which no enhanced service agreement exists may reset the DS codepoint to CS0. As a consequence, an interconnect SLA needs to specify not only the treatment of traffic that arrives with a supported interconnect DSCP, but also the treatment of traffic that arrives with unsupported or unexpected DSCPs; remarking to CS0 is a widely deployed behavior.

During the process of setting up a Diffserv interconnection, both networks should define the set of acceptable and unacceptable DSCPs and specify the treatment of traffic marked with each DSCP.

While Diffserv-Intercon allows modification of unacceptable DSCPs, if traffic using one or more of the PHBs in a PHB group (e.g., AF3x, consisting of AF31, AF32 and AF33) is accepted as part of a supported Diffserv-Intercon Treatment Aggregate, then traffic using other PHBs from the same PHB group should not be modified to use PHBs outside of that PHB group, and in particular should not be remarked to CS0 unless the entire PHB group is remarked to CS0. This avoids unexpected forwarding behavior (and potential reordering, see also [RFC7657]) when using Assured Forwarding (AF) PHBs [RFC2597].

4.1. Diffserv-Intercon Example

The overall approach to DSCP marking at network interconnections is illustrated by the following example. Provider O and provider W are peered with provider T. They have agreed upon a Diffserv interconnection SLA.

Traffic of provider O terminates within provider T’s network, while provider W’s traffic transits through the network of provider T to provider F. This example assumes that all providers use their own internal PHB and codepoint (DSCP) that correspond to the AF31 PHB in

the Diffserv-Intercon Assured Elastic Treatment Aggregate (AF21 and CS2 are used in the example).

```
    Provider O            Provider W
          |                     |
     +----------+          +----------+
     |  AF21    |          |    CS2   |
     +----------+          +----------+
          V                     V
      +~~~~~~~+             +~~~~~~~+
      |Rtr PrO|             |Rtr PrW|              Rtr:   Router
      +~~~~~~~+             +~~~~~~~+            Pr[L]:   Provider[L]
          |        DiffServ      |
     +----------+          +----------+
     |    AF31  |          |    AF31  |
     +----------+          +----------+
          V        Intercon      V
      +~~~~~~~+                 |
      |RtrPrTI|-----------------+           Router Provider T Ingress
      +~~~~~~~+
          |            Provider T domain
     +-----------------+
     |  MPLS TC 2, AF21 |
     +-----------------+
        |      |    +----------+   +~~~~~~~+
        V      '--->|   AF21   |->-|RtrDstH|    Router Destination Host
    +----------+    +----------+   +~~~~~~~+
    |   AF21   |       Local DSCPs Provider T
    +----------+
        |
     +~~~~~~~+
     |RtrPrTE|                              Router Provider T Egress
     +~~~~~~~+
        |        DiffServ
    +----------+
    |    AF31  |
    +----------+
        |        Intercon
     +~~~~~~~+
     |RtrPrF |                              Router Provider F
     +~~~~~~~+
        |
    +----------+
    |   AF11   |    Provider F
    +----------+
```

Diffserv-Intercon example

Figure 1

Providers only need to deploy mappings of internal DSCPs to/from Diffserv-Intercon DSCPs so that they can exchange traffic using the desired PHBs. In the example, provider O has decided that the properties of his internal class AF21 are best met by the Diffserv-Intercon Assured Elastic Treatment Aggregate, PHB AF31. At the outgoing peering interface connecting provider O with provider T the former's peering router remarks AF21 traffic to AF31. The domain internal PHB of provider T meeting the requirement of Diffserv-Intercon Assured Elastic Treatment Aggregate are from AF2x PHB group. Hence AF31 traffic received at the interconnection with provider T is remarked to AF21 by the peering router of domain T, and domain T has chosen to use MPLS Traffic Class value 2 for this aggregate. At the penultimate MPLS node, the top MPLS label is removed and exposes the IP header marked by the DSCP that has been set at the network ingress. The peering router connecting domain T with domain F classifies the packet by its domain-T-internal DSCP AF21. As the packet leaves domain T on the interface to domain F, this causes the packet's DSCP to be remarked to AF31. The peering router of domain F classifies the packet for domain-F-internal PHB AF11, as this is the PHB with properties matching Diffserv-Intercon's Assured Elastic Treatment Aggregate.

This example can be extended. The figure shows Provider-W using CS2 for traffic that corresponds to Diffserv-Intercon Assured Elastic Treatment Aggregate PHB AF31; that traffic is mapped to AF31 at the Diffserv-Intercon interconnection to Provider-T. In addition, suppose that Provider-O supports a PHB marked by AF22 and this PHB is supposed to obtain Diffserv transport within Provider-T domain. Then Provider-O will remark it with DSCP AF32 for interconnection to Provider-T.

Finally suppose that Provider-W supports CS3 for internal use only. Then no Diffserv- Intercon DSCP mapping needs to be configured at the peering router. Traffic, sent by Provider-W to Provider-T marked by CS3 due to a misconfiguration may be remarked to CS0 by Provider-T.

4.2. End-to-end PHB and DSCP Transparency

This section briefly discusses end-to-end Diffserv approaches related to the Uniform, Pipe and Short Pipe tunnel models ([RFC2983], [RFC3270]), when used edge-to-edge in a network.

- o With the Uniform model, neither the DCSP nor the PHB change. This implies that a network management packet received with a CS6 DSCP would be forwarded with an MPLS Traffic Class corresponding to CS6. The uniform model is outside the scope of this document.

- o With the Pipe model, the inner tunnel DCSP remains unchanged, but an outer tunnel DSCP and the PHB could changed. For example a packet received with a (network specific) CS1 DSCP would be transported by default PHB and if MPLS is applicable, forwarded with an MPLS Traffic Class corresponding to Default PHB. The CS1 DSCP is not rewritten. Transport of a large variety (much greater than 4) DSCPs may be required across an interconnected network operating MPLS Short pipe transport for IP traffic. In that case, a tunnel based on the Pipe model is among the possible approaches. The Pipe model is outside the scope of this document.

- o With the Short Pipe model, the DCSP likely changes and the PHB might change. This document describes a method to simplify Diffserv network interconnection when a DSCP rewrite can’t be avoided.

4.3. Treatment of Network Control traffic at carrier interconnection interfaces

As specified by RFC4594, section 3.2, Network Control (NC) traffic marked by CS6 is expected at some interconnection interfaces. This document does not change RFC4594, but observes that network control traffic received at network ingress is generally different from network control traffic within a network that is the primary use of CS6 envisioned by RFC4594. A specific example is that some CS6 traffic exchanged across carrier interconnections is terminated at the network ingress node, e.g., when BGP is used between the two routers on opposite ends of an interconnection link; in this case the operators would enter into a bilateral agreement to use CS6 for that BGP traffic.

The end-to-end discussion in the previous section (4.2) is generally inapplicable to network control traffic - network control traffic is generally intended to control a network, not be transported between networks. One exception is that network control traffic makes sense for a purchased transit agreement, and preservation of the CS6 DSCP marking for network control traffic that is transited is reasonable in some cases, although it is generally inappropriate to use CS6 for forwarding that traffic within the network that provides transit. Use of an IP tunnel is suggested in order to conceal the CS6 markings on transiting network control traffic from the network that provides the transit. In this case, Pipe model for Diffserv tunneling is used.

If the MPLS Short Pipe model is deployed for unencapsulated IPv4 traffic, an IP network provider should limit access to the CS6 and CS7 DSCPs so that they are only used for network control traffic for the provider’s own network.

Interconnecting carriers should specify treatment of CS6 marked traffic received at a carrier interconnection which is to be forwarded beyond the ingress node. An SLA covering the following cases is recommended when a provider wishes to send CS6 marked traffic across an interconnection link and that traffic's destination is beyond the interconnected ingress node:

- o classification of traffic that is network control traffic for both domains. This traffic should be classified and marked for the CS6 DSCP.

- o classification of traffic that is network control traffic for the sending domain only. This traffic should be forwarded with a PHB that is appropriate for the NC service class [RFC4594], e.g., AF31 as specified by this document. As an example GSMA IR.34 recommends an Interactive class / AF31 to carry SIP and DIAMETER traffic. While this is service control traffic of high importance to interconnected Mobile Network Operators, it is certainly not Network Control traffic for a fixed network providing transit among such operators, and hence should not receive CS6 treatment in such a transit network.

- o any other CS6 marked traffic should be remarked or dropped.

5. Acknowledgements

Bob Briscoe and Gorry Fairhurst reviewed the draft and provided rich feedback. Brian Carpenter, Fred Baker, Al Morton and Sebastien Jobert discussed the draft and helped improving it. Mohamed Boucadair and Thomas Knoll helped adding awareness of related work. James Polk's discussion during IETF 89 helped to improve the text on the relation of this draft to RFC4594 and RFC5127.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

The DSCP field in the IP header can expose additional traffic classification information at network interconnections by comparison to use of a zero DSCP for all interconnect traffic. If traffic classification info is sensitive, the DSCP field could be remarked to zero to hide the classification as a countermeasure, at the cost of loss of Diffserv info and differentiated traffic handling on the interconnect and subsequent networks. When AF PHBs are used, any such remarking should respect AF PHB group boundaries as further discussed at the end of Section 4.

This document does not introduce new features; it describes how to use existing ones. The Diffserv security considerations in [RFC2475] and [RFC4594] apply.

8. References

8.1. Normative References

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <http://www.rfc-editor.org/info/rfc2474>.

[RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, DOI 10.17487/RFC2597, June 1999, <http://www.rfc-editor.org/info/rfc2597>.

[RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, DOI 10.17487/RFC3246, March 2002, <http://www.rfc-editor.org/info/rfc3246>.

[RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <http://www.rfc-editor.org/info/rfc3270>.

[RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <http://www.rfc-editor.org/info/rfc5129>.

[RFC5865] Baker, F., Polk, J., and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, DOI 10.17487/RFC5865, May 2010, <http://www.rfc-editor.org/info/rfc5865>.

8.2. Informative References

[I-D.knoll-idr-cos-interconnect] Knoll, T., "BGP Class of Service Interconnection", draft-knoll-idr-cos-interconnect-17 (work in progress), November 2016.

[ID.ietf-idr-sla]
IETF, "Inter-domain SLA Exchange", IETF, http://datatracker.ietf.org/doc/draft-ietf-idr-sla-exchange/, 2013.

[IR.34] GSMA Association, "IR.34 Inter-Service Provider IP Backbone Guidelines Version 7.0", GSMA, GSMA IR.34 http://www.gsma.com/newsroom/wp-content/uploads/2012/03/ir.34.pdf, 2012.

[MEF23.1] MEF, "Implementation Agreement MEF 23.1 Carrier Ethernet Class of Service Phase 2", MEF, MEF23.1 http://metroethernetforum.org/PDF_Documents/technical-specifications/MEF_23.1.pdf, 2012.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <http://www.rfc-editor.org/info/rfc2475>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <http://www.rfc-editor.org/info/rfc2983>.

[RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services", RFC 3662, DOI 10.17487/RFC3662, December 2003, <http://www.rfc-editor.org/info/rfc3662>.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <http://www.rfc-editor.org/info/rfc4594>.

[RFC5127] Chan, K., Babiarz, J., and F. Baker, "Aggregation of Diffserv Service Classes", RFC 5127, DOI 10.17487/RFC5127, February 2008, <http://www.rfc-editor.org/info/rfc5127>.

[RFC5160] Levis, P. and M. Boucadair, "Considerations of Provider-to-Provider Agreements for Internet-Scale Quality of Service (QoS)", RFC 5160, DOI 10.17487/RFC5160, March 2008, <http://www.rfc-editor.org/info/rfc5160>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <http://www.rfc-editor.org/info/rfc7657>.

[Y.1566] ITU-T, "Quality of service mapping and interconnection between Ethernet, IP and multiprotocol label switching networks", ITU, http://www.itu.int/rec/T-REC-Y.1566-201207-I/en, 2012.

Appendix A. Appendix A The MPLS Short Pipe Model and IP traffic

The MPLS Short Pipe Model (or penultimate Hop Label Popping) is widely deployed in carrier networks. If unencapsulated IP traffic is transported using MPLS Short Pipe, IP headers appear inside the last section of the MPLS domain. This impacts the number of PHBs and DSCPs that a network provider can reasonably support. See Figure 2 (below) for an example.

For encapsulated IP traffic, only the outer tunnel header is relevant for forwarding. If the tunnel does not terminate within the MPLS network section, only the outer tunnel DSCP is involved, as the inner DSCP does not affect forwarding behavior; in this case all DSCPs could be used in the inner IP header without affecting network behavior based on the outer MPLS header. Here the Pipe model applies.

Layer 2 and Layer 3 VPN traffic all use an additional MPLS label; in this case, the MPLS tunnel follows the Pipe model. Classification and queuing within an MPLS network is always based on an MPLS label, as opposed to the outer IP header.

Carriers often select PHBs and DSCP without regard to interconnection. As a result PHBs and DSCPs typically differ between network carriers. With the exception of best effort traffic, a DSCP change should be expected at an interconnection at least for unencapsulated IP traffic, even if the PHB is suitably mapped by the carriers involved.

Although RFC3270 suggests that the Short Pipe Model is only applicable to VPNs, current networks also use it to transport non-tunneled IPv4 traffic. This is shown in figure 2 where Diffserv-Intercon is not used, resulting in exposure of the internal DSCPs of the upstream network to the downstream network across the interconnection.

```
        |
       \|/           IPv4, DSCP_send
        V
        |
   Peering Router
        |
       \|/           IPv4, DSCP_send
        V
        |
   MPLS Edge Router
        |          Mark MPLS Label, TC_internal
       \|/         Remark DSCP to
        V            (Inner: IPv4, DSCP_d)
        |
   MPLS Core Router  (penultimate hop label popping)
        |                         \
        |            IPv4, DSCP_d |  The DSCP needs to be in network-
        |                 ^^^^^^^^|  internal Diffserv context. The Core
       \|/                         > Router may require or enforce
        V                         |  that. The Edge Router may wrongly
        |                         |  classify, if the DSCP is not in
        |                        /   network-internal Diffserv context.
   MPLS Edge Router
        |                         \   Traffic leaves the network marked
       \|/           IPv4, DSCP_d |  with the network-internal
        V                         > DSCP_d that must be dealt with
        |                         |  by the next network (downstream).
        |                        /
   Peer Router
        |          Remark DSCP to
       \|/           IPv4, DSCP_send
        V
        |


   Short-Pipe / penultimate hop popping example
```

Figure 2

The packets IP DSCP must be in a well understood Diffserv context for schedulers and classifiers on the interfaces of the ultimate MPLS link (last link traversed before leaving the network). The necessary Diffserv context is network-internal and a network operating in this mode enforces DSCP usage in order to obtain robust differentiated forwarding behavior.

Without Diffserv-Intercon treatment, the traffic is likely to leave each network marked with network-internal DSCP. DSCP_send in the

figure above has to be remarked into the first network's Diffserv scheme at the ingress MPLS Edge Router, to DSCP_d in the example. For that reason, the traffic leaves this domain marked by the network-internal DSCP_d. This structure requires that every carrier deploys per-peer PHB and DSCP mapping schemes.

If Diffserv-Intercon is applied DSCPs for traffic transiting the domain can be mapped from and remapped to an original DSCP. This is shown in figure 3. Internal traffic may continue to use internal DSCPs (e.g., DSCP_d) and they may also be used between a carrier and its direct customers.

```
   Internal Router
        |
        |   Outer Header
       \|/    IPv4, DSCP_send
        V
        |
   Peering Router
        |  Remark DSCP to
       \|/    IPv4, DSCP_ds-int    Diffserv-Intercon DSCP and PHB
        V
        |
   MPLS Edge Router
        |
        |   Mark  MPLS Label, TC_internal
       \|/  Remark DSCP to
        V     (Inner: IPv4, DSCP_d)   domain internal DSCP for
        |                             the PHB
   MPLS Core Router  (penultimate hop label popping)
        |
        |     IPv4, DSCP_d
        |           ^^^^^^
       \|/
        V
        |
        |
   MPLS Edge Router--------------------+
        |                              |
       \|/  Remark DSCP to            \|/  IPv4, DSCP_d
        V     IPv4, DSCP_ds-int        V
        |                              |
        |                              |
   Peer Router              Domain internal Broadband
        |                        Access Router
       \|/  Remark DSCP to            \|/
        V     IPv4, DSCP_send          V  IPv4, DSCP_d
        |                              |
```

Short-Pipe example with Diffserv-Intercon

Figure 3

Authors' Addresses

Ruediger Geib (editor)
Deutsche Telekom
Heinrich Hertz Str. 3-7
Darmstadt 64295
Germany

Phone: +49 6151 5812747
Email: Ruediger.Geib@telekom.de

David L. Black
Dell EMC
176 South Street
Hopkinton, MA
USA

Phone: +1 (508) 293-7953
Email: david.black@dell.com

Transport Area Working Group
Internet-Draft
Updates: 3819 (if approved)
Intended status: Best Current Practice
Expires: 7 June 2024

B. Briscoe
Independent
J. Kaippallimalil
Futurewei

5 December 2023

Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP

draft-ietf-tsvwg-ecn-encap-guidelines-22

Abstract

The purpose of this document is to guide the design of congestion notification in any lower layer or tunnelling protocol that encapsulates IP. The aim is for explicit congestion signals to propagate consistently from lower layer protocols into IP. Then the IP internetwork layer can act as a portability layer to carry congestion notification from non-IP-aware congested nodes up to the transport layer (L4). Following these guidelines should assure interworking among IP layer and lower layer congestion notification mechanisms, whether specified by the IETF or other standards bodies. This document is included in BCP 89 and updates the single paragraph of advice to subnetwork designers about ECN in Section 13 of RFC 3819, by replacing it with a reference to the whole of this document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The benefits of Explicit Congestion Notification (ECN) described in [RFC8087] and summarized below can only be fully realized if support for ECN is added to the relevant subnetwork technology, as well as to IP. When a lower layer buffer drops a packet obviously it does not just drop at that layer; the packet disappears from all layers. In contrast, when active queue management (AQM) at a lower layer marks a packet with ECN, the marking needs to be explicitly propagated up the layers. The same is true if AQM marks the outer header of a packet that encapsulates inner tunnelled headers. Forwarding ECN is not as straightforward as other headers because it has to be assumed ECN may be only partially deployed. If a lower layer header that contains ECN congestion indications is stripped off by a subnet egress that is not ECN-aware, or if the ultimate receiver or sender is not ECN-aware, congestion needs to be indicated by dropping the packet, not marking it.

The purpose of this document is to guide the addition of congestion notification to any subnet technology or tunnelling protocol, so that lower layer AQM algorithms can signal congestion explicitly and it will propagate consistently into encapsulated (higher layer) headers, otherwise the signals will not reach their ultimate destination.

ECN is defined in the IP header (IPv4 and IPv6) [RFC3168] to allow a resource to notify the onset of queue build-up without having to drop packets, by explicitly marking a proportion of packets with the congestion experienced (CE) codepoint.

Given a suitable marking scheme, ECN removes nearly all congestion loss and it cuts delays for two main reasons:

* It avoids the delay when recovering from congestion losses, which particularly benefits small flows or real-time flows, making their delivery time predictably short [RFC2884];

* As ECN is used more widely by end-systems, it will gradually remove the need to configure a degree of delay into buffers before they start to notify congestion (the cause of bufferbloat). This is because drop involves a trade-off between sending a timely signal and trying to avoid impairment, whereas ECN is solely a signal not an impairment, so there is no harm triggering it earlier.

Some lower layer technologies (e.g. MPLS, Ethernet) are used to form subnetworks with IP-aware nodes only at the edges. These networks are often sized so that it is rare for interior queues to overflow. However, until recently this was more due to the inability of TCP to saturate the links. For many years, fixes such as window scaling [RFC7323] proved hard to deploy. And the Reno variant of TCP has remained in widespread use despite its inability to scale to high flow rates. However, now that modern operating systems are finally capable of saturating interior links, even the buffers of well-provisioned interior switches will need to signal episodes of queuing.

Propagation of ECN is defined for MPLS [RFC5129], and has been defined for TRILL [RFC7780], [I-D.ietf-trill-ecn-support], but it remains to be defined for a number of other subnetwork technologies.

Similarly, ECN propagation is yet to be defined for many tunnelling protocols. [RFC6040] defines how ECN should be propagated for IP-in-IPv4 [RFC2003], IP-in-IPv6 [RFC2473] and IPsec [RFC4301] tunnels, but there are numerous other tunnelling protocols with a shim and/or a layer 2 header between two IP headers (IPv4 or IPv6). Some address ECN propagation between the IP headers, but many do not. This document gives guidance on how to address ECN propagation for future tunnelling protocols, and a companion standards track specification [I-D.ietf-tsvwg-rfc6040update-shim] updates those existing IP-shim-(L2)-IP protocols that are under IETF change control and still widely used.

Incremental deployment is the most delicate aspect when adding support for ECN. The original ECN protocol in IP [RFC3168] was carefully designed so that a congested buffer would not mark a packet (rather than drop it) unless both source and destination hosts were ECN-capable. Otherwise, its congestion markings would never be detected and congestion would just build up further. However, to support congestion marking below the IP layer or within tunnels, it is not sufficient to only check that the two layer 4 transport end-points support ECN; correct operation also depends on the decapsulator at each subnet or tunnel egress faithfully propagating congestion notifications to the higher layer. Otherwise, a legacy decapsulator might silently fail to propagate any ECN signals from

the outer to the forwarded header. Then the lost signals would never be detected and again congestion would build up further. The guidelines given later require protocol designers to carefully consider incremental deployment, and suggest various safe approaches for different circumstances.

Of course, the IETF does not have standards authority over every link layer protocol. So this document gives guidelines for designing propagation of congestion notification across the interface between IP and protocols that may encapsulate IP (i.e. that can be layered beneath IP). Each lower layer technology will exhibit different issues and compromises, so the IETF or the relevant standards body must be free to define the specifics of each lower layer congestion notification scheme. Nonetheless, if the guidelines are followed, congestion notification should interwork between different technologies, using IP in its role as a 'portability layer'.

Therefore, the capitalized terms 'SHOULD' or 'SHOULD NOT' are often used in preference to 'MUST' or 'MUST NOT', because it is difficult to know the compromises that will be necessary in each protocol design. If a particular protocol design chooses not to follow a 'SHOULD (NOT)' given in the advice below, it MUST include a sound justification.

It has not been possible to give common guidelines for all lower layer technologies, because they do not all fit a common pattern. Instead, they have been divided into a few distinct modes of operation: feed-forward-and-upward; feed-upward-and-forward; feed-backward; and null mode. These modes are described in Section 3, then in the subsequent sections separate guidelines are given for each mode.

1.1. Update to RFC 3819

This document updates the brief advice to subnetwork designers about ECN in Section 13 of [RFC3819], by replacing the last two paragraphs with the following sentence:

> By following the guidelines in [this document], subnetwork designers can enable a layer-2 protocol to participate in congestion control without dropping packets via propagation of explicit congestion notification (ECN [RFC3168]) to receivers.

and adding [this document] as an informative reference. {RFC Editor: Please replace both instances of [this document] above with the number of the present RFC when published.}

1.2. Scope

This document only concerns wire protocol processing of explicit notification of congestion. It makes no changes or recommendations concerning algorithms for congestion marking or for congestion response, because algorithm issues should be independent of the layer the algorithm operates in.

The default ECN semantics are described in [RFC3168] and updated by [RFC8311]. Also, the guidelines for AQM designers [RFC7567] clarify the semantics of both drop and ECN signals from AQM algorithms. [RFC4774] is the appropriate best current practice specification of how algorithms with alternative semantics for the ECN field can be partitioned from Internet traffic that uses the default ECN semantics. There are two main examples for how alternative ECN semantics have been defined in practice:

* RFC 4774 suggests using the ECN field in combination with a Diffserv codepoint such as in PCN [RFC6660], Voice over 3G [UTRAN] or Voice over LTE (VoLTE) [LTE-RA];

* RFC 8311 suggests using the ECT(1) codepoint of the ECN field to indicate alternative semantics such as for the experimental Low Latency Low Loss Scalable throughput (L4S) service [RFC9331]).

The aim is that the default rules for encapsulating and decapsulating the ECN field are sufficiently generic that tunnels and subnets will encapsulate and decapsulate packets without regard to how algorithms elsewhere are setting or interpreting the semantics of the ECN field. [RFC6040] updates RFC 4774 to allow alternative encapsulation and decapsulation behaviours to be defined for alternative ECN semantics. However, it reinforces the same point - that it is far preferable to try to fit within the common ECN encapsulation and decapsulation behaviours, because expecting all lower layer technologies and tunnels to be updated is likely to be completely impractical.

Alternative semantics for the ECN field can be defined to depend on the traffic class indicated by the DSCP. Therefore, correct propagation of congestion signals could depend on correct propagation of the DSCP between the layers and along the path. For instance, if the meaning of the ECN field depends on the DSCP (as in PCN or VoLTE) and if the outer DSCP is stripped on descapsulation, as in the pipe model of [RFC2983], the special semantics of the ECN field would be lost. Similarly, if the DSCP is changed at the boundary between Diffserv domains, the special ECN semantics would also be lost. This is an important implication of the localized scope of most Diffserv arrangements. In this document, correct propagation of traffic class information is assumed, while what 'correct' means and how it is achieved is covered elsewhere (e.g. RFC 2983) and is outside the scope of the present document.

The guidelines in this document do ensure that common encapsulation and decapsulation rules are sufficiently generic to cover cases where ECT(1) is used instead of ECT(0) to identify alternative ECN semantics (as in L4S [RFC9331]) and where ECN marking algorithms use ECT(1) to encode 3 severity levels into the ECN field (e.g. PCN [RFC6660]) rather than the default of 2. All these different semantics for the ECN field work because it has been possible to define common default decapsulation rules that allow for all cases.

Note that the guidelines in this document do not necessarily require the subnet wire protocol to be changed to add support for congestion notification. For instance, the Feed-Up-and-Forward Mode (Section 3.2) and the Null Mode (Section 3.4) do not. Another way to add congestion notification without consuming header space in the subnet protocol might be to use a parallel control plane protocol.

This document focuses on the congestion notification interface between IP and lower layer or tunnel protocols that can encapsulate IP, where the term 'IP' includes IPv4 or IPv6, unicast, multicast or anycast. However, it is likely that the guidelines will also be useful when a lower layer protocol or tunnel encapsulates itself, e.g. Ethernet MAC in MAC ([IEEE802.1Q]; previously 802.1ah) or when it encapsulates other protocols. In the feed-backward mode, propagation of congestion signals for multicast and anycast packets is out-of-scope (because the complexity would make it unlikely to be attempted).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Further terminology used within this document:

Protocol data unit (PDU): Information that is delivered as a unit among peer entities of a layered network consisting of protocol control information (typically a header) and possibly user data (payload) of that layer. The scope of this document includes layer 2 and layer 3 networks, where the PDU is respectively termed a frame or a packet (or a cell in ATM). PDU is a general term for any of these. This definition also includes a payload with a shim header lying somewhere between layer 2 and 3.

Transport: The end-to-end transmission control function, conventionally considered at layer-4 in the OSI reference model. Given the audience for this document will often use the word transport to mean low level bit carriage, whenever the term is used it will be qualified, e.g. 'L4 transport'.

Encapsulator: The link or tunnel endpoint function that adds an outer header to a PDU (also termed the 'link ingress', the 'subnet ingress', the 'ingress tunnel endpoint' or just the 'ingress' where the context is clear).

Decapsulator: The link or tunnel endpoint function that removes an outer header from a PDU (also termed the 'link egress', the 'subnet egress', the 'egress tunnel endpoint' or just the 'egress' where the context is clear).

Incoming header: The header of an arriving PDU before encapsulation.

Outer header: The header added to encapsulate a PDU.

Inner header: The header encapsulated by the outer header.

Outgoing header: The header forwarded by the decapsulator.

CE: Congestion Experienced [RFC3168]

ECT: ECN-Capable (L4) Transport [RFC3168]

Not-ECT: Not ECN-Capable (L4) Transport [RFC3168]

Load Regulator: For each flow of PDUs, the transport function that is capable of controlling the data rate. Typically located at the data source, but in-path nodes can regulate load in some congestion control arrangements (e.g. admission control, policing nodes or transport circuit-breakers [RFC8084]). Note the term "a function capable of controlling the load" deliberately includes a transport that does not actually control the load responsively but ideally it ought to (e.g. a sending application without congestion control that uses UDP).

ECN-PDU: A PDU at the IP layer or below with a capacity to signal congestion that is part of a congestion control feedback loop within which all the nodes necessary to propagate the signal back to the Load Regulator are capable of doing that propagation. An IP packet with a non-zero ECN field implies that the endpoints are ECN-capable, so this would be an ECN-PDU. However, ECN-PDU is intended to be a general term for a PDU at lower layers, as well as at the IP layer.

Not-ECN-PDU: A PDU at the IP layer or below that is part of a congestion control feedback loop that is not capable of propagating explicit congestion notification signals back to the Load Regulator, because at least one of the nodes necessary to propagate the signals is incapable of doing that propagation. Note that this definition is a property of the feedback-loop, not necessarily of the PDU itself, because in some protocols the PDU will self-describe the property, but in others the property might be carried in a separate control-plane context that is somehow bound to the PDU.

3. Modes of Operation

This section sets down the different modes by which congestion information is passed between the lower layer and the higher one. It acts as a reference framework for the following sections, which give normative guidelines for designers of explicit congestion notification protocols, taking each mode in turn:

Feed-Forward-and-Up: Nodes feed forward congestion notification towards the egress within the lower layer then up and along the layers towards the end-to-end destination at the transport layer. The following local optimisation is possible:

Feed-Up-and-Forward: A lower layer switch feeds-up congestion notification directly into the higher layer (e.g. into the ECN field in the IP header), irrespective of whether the node is at the egress of a subnet.

Feed-Backward: Nodes feed back congestion signals towards the ingress of the lower layer and (optionally) attempt to control congestion within their own layer.

Null: Nodes cannot experience congestion at the lower layer except at ingress nodes (which are IP-aware or equivalently higher-layer-aware).

3.1. Feed-Forward-and-Up Mode

Like IP and MPLS, many subnet technologies are based on self-contained protocol data units (PDUs) or frames sent unreliably. They provide no feedback channel at the subnetwork layer, instead relying on higher layers (e.g. TCP) to feed back loss signals.

In these cases, ECN may best be supported by standardising explicit notification of congestion into the lower layer protocol that carries the data forwards. Then a specification is needed for how the egress of the lower layer subnet propagates this explicit signal into the forwarded upper layer (IP) header. This signal continues forwards until it finally reaches the destination transport (at L4). Then typically the destination will feed this congestion notification back to the source transport using an end-to-end protocol (e.g. TCP). This is the arrangement that has already been used to add ECN to IP-in-IP tunnels [RFC6040], IP-in-MPLS and MPLS-in-MPLS [RFC5129].

This mode is illustrated in Figure 1. Along the middle of the figure, layers 2, 3 and 4 of the protocol stack are shown, and one packet is shown along the bottom as it progresses across the network from source to destination, crossing two subnets connected by a router, and crossing two switches on the path across each subnet. Congestion at the output of the first switch (shown as *) leads to a congestion marking in the L2 header (shown as C in the illustration of the packet). The chevrons show the progress of the resulting congestion indication. It is propagated from link to link across the subnet in the L2 header, then when the router removes the marked L2 header, it propagates the marking up into the L3 (IP) header. The router forwards the marked L3 header into subnet B. The L2 protocol used in subnet B does not support ECN, but the signal proceeds across it in the L3 header.

Note that there is no implication that each ’C’ marking is encoded the same; a different encoding might be used for the ’C’ marking in each protocol.

Finally, for completeness, we show the L3 marking arriving at the destination, where the host transport protocol (e.g. TCP) feeds it back to the source in the L4 acknowledgement (the 'C' at L4 in the packet at the top of the diagram).

```
                        _ _ _
          /_______     |_|_|C|  ACK Packet (V)
          \            |_|_|_|
 +---+        layer: 2 3 4 header                            +---+
 |  <|<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Packet V <<<<<<<<<<<<<|<< |L4
 |   |                            +---+                      | ^ |
 |   | . . . . . . Packet U. . | >>|>>> Packet U >>>>>>>>>>>>|>^ |L3
 |   |     +---+     +---+      | ^ |     +---+     +---+   |   |
 |   |     |  *|>>>>>|>>>|>>>>>|>^ |     |   |     |   |   |   |L2
 |___|_____|___|_____|___|_____|___|_____|___|_____|___|_____|___|
 source          subnet A      router       subnet B         dest
   __ _ _ _|   __ _ _ _|  __ _ _|       __ _ _ _|
  |__|_|_|_|  |__|_|_|C| |__|_|C|      |__|_|C|_|    Data________\
  |__|_|_|_|  |__|_|_|_| |__|_|_|      |__|_|_|_|    Packet (U)  /
 layer:4 3 2A      4 3 2A     4 3           4 3 2B
 header
```

Figure 1: Feed-Forward-and-Up Mode

Of course, modern networks are rarely as simple as this text-book example, often involving multiple nested layers. For example, a 3GPP mobile network may have two IP-in-IP (GTP [GTPv1]) tunnels in series and an MPLS backhaul between the base station and the first router. Nonetheless, the example illustrates the general idea of feeding congestion notification forward then upward whenever a header is removed at the egress of a subnet.

Note that the FECN (forward ECN) bit in Frame Relay [Buck00] and the explicit forward congestion indication (EFCI [ITU-T.I.371]) bit in ATM user data cells follow a feed-forward pattern. However, in ATM, this arrangement is only part of a feed-forward-and-backward pattern at the lower layer, not feed-forward-and-up out of the lower layer the intention was never to interface to IP ECN at the subnet egress. To our knowledge, Frame Relay FECN is solely used to detect where more capacity should be provisioned.

3.2. Feed-Up-and-Forward Mode

Ethernet is particularly difficult to extend incrementally to support explicit congestion notification. One way to support ECN in such cases has been to use so called 'layer-3 switches'. These are Ethernet switches that dig into the Ethernet payload to find an IP header and manipulate or act on certain IP fields (specifically Diffserv & ECN). For instance, in Data Center TCP [RFC8257], layer-3 switches are configured to mark the ECN field of the IP header within the Ethernet payload when their output buffer becomes congested. With respect to switching, a layer-3 switch acts solely on the addresses in the Ethernet header; it does not use IP addresses, and it does not decrement the TTL field in the IP header.

```
                    _ _ _
          /_______  | | |C|  ACK packet (V)
          \         |_|_|_|
 +---+        layer: 2 3 4 header                          +---+
 |  <|<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Packet V <<<<<<<<<<<<<|<< |L4
 |   |                           +---+                      | ^ |
 |   | . . .  >>>> Packet U >>>|>>>|>>> Packet U >>>>>>>>>>>>|>^ |L3
 |   |      +--^+     +---+      | v|      +---+     +---+  | ^ |
 |   |      |  *|     |   |      | >|>>>>>|>>>|>>>>>|>>>|>>>>>|>^ |L2
 |___|______|___|_____|___|______|__|______|___|_____|___|_____|___|
 source          subnet E      router       subnet F         dest
   __ _ _ _|   __ _ _ _|  __ _ _|       __ _ _ _|
  | | | | |  | | |C| |  | | |C|       | | |C|C|    Data________\
  |_|_|_|_|  |_|_|_|_|  |_|_|_|       |_|_|_|_|    Packet (U)   /
 layer:4 3 2        4 3 2       4 3            4 3 2
 header
```

Figure 2: Feed-Up-and-Forward Mode

By comparing Figure 2 with Figure 1, it can be seen that subnet E (perhaps a subnet of layer-3 Ethernet switches) works in feed-up-and-forward mode by notifying congestion directly into L3 at the point of congestion, even though the congested switch does not otherwise act at L3. In this example, the technology in subnet F (e.g. MPLS) does support ECN natively, so when the router adds the layer-2 header it copies the ECN marking from L3 to L2 as well, as shown by the 'C's in both layers.

3.3. Feed-Backward Mode

In some layer 2 technologies, explicit congestion notification has been defined for use internally within the subnet with its own feedback and load regulation, but typically the interface with IP for ECN has not been defined.

For instance, for the available bit-rate (ABR) service in ATM, the relative rate mechanism was one of the more popular mechanisms for managing traffic, tending to supersede earlier designs. In this approach ATM switches send special resource management (RM) cells in both the forward and backward directions to control the ingress rate of user data into a virtual circuit. If a switch buffer is approaching congestion or is congested it sends an RM cell back towards the ingress with respectively the No Increase (NI) or Congestion Indication (CI) bit set in its message type field [ATM-TM-ABR]. The ingress then holds or decreases its sending bit-rate accordingly.

```
                          _ _ _
          /_______  |_|_|C|  ACK packet (X)
          \         |_|_|_|
 +---+        layer: 2 3 4 header                              +---+
 |  <|<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Packet X <<<<<<<<<<<<<|<<  |L4
 |   |                          +---+                        |  ^  |
 |   |                          |  *|>>> Packet W >>>>>>>>>>>|>^  |L3
 |   |     +---+     +---+      |   |     +---+     +---+    |   |
 |   |     |   |     |   |      |  <|<<<<<|<<<|<(V)<|<<<|    |   |L2
 |___|_____|___|_____|___|_____|___|_____|___|_____|___|_____|___|L2
 source          subnet G      router       subnet H         dest
    __ _ _ _   __ _ _ _   __ _ _        __ _ _ _   later
   |_|_|_|_|  |_|_|_|_|  |_|_|_|       |_|_|C|_|  data________\
   |_|_|_|_|  |_|_|_|_|  |_|_|_|       |_|_|_|_|  packet (W)  /
       4 3 2       4 3 2       4 3           4 3 2

                                   _
                                /__ |C|  Feedback control
                                \   |_|  cell/frame (V)
                                     2
    __ _ _ _   __ _ _ _   __ _ _        __ _ _ _   earlier
   |_|_|_|_|  |_|_|_|_|  |_|_|_|       |_|_|_|_|  data________\
   |_|_|_|_|  |_|_|_|_|  |_|_|_|       |_|_|_|_|  packet (U)  /
layer:  4 3 2       4 3 2       4 3           4 3 2
header
```

Figure 3: Feed-Backward Mode

ATM's feed-backward approach does not fit well when layered beneath IP's feed-forward approach unless the initial data source is the same node as the ATM ingress. Figure 3 shows the feed-backward approach being used in subnet H. If the final switch on the path is congested (*), it does not feed-forward any congestion indications on packet (U). Instead it sends a control cell (V) back to the router at the ATM ingress.

However, the backward feedback does not reach the original data source directly because IP does not support backward feedback (and subnet G is independent of subnet H). Instead, the router in the middle throttles down its sending rate but the original data sources don't reduce their rates. The resulting rate mismatch causes the middle router's buffer at layer 3 to back up until it becomes congested, which it signals forwards on later data packets at layer 3 (e.g. packet W). Note that the forward signal from the middle router is not triggered directly by the backward signal. Rather, it is triggered by congestion resulting from the middle router's mismatched rate response to the backward signal.

In response to this later forward signalling, end-to-end feedback at layer-4 finally completes the tortuous path of congestion indications back to the origin data source, as before.

Quantized congestion notification (QCN [IEEE802.1Q]) would suffer from similar problems if extended to multiple subnets. However, from the start QCN was clearly characterized as solely applicable to a single subnet (see Section 6).

3.4. Null Mode

Often link and physical layer resources are 'non-blocking' by design. In these cases congestion notification may be implemented but it does not need to be deployed at the lower layer; ECN in IP would be sufficient.

A degenerate example is a point-to-point Ethernet link. Excess loading of the link merely causes the queue from the higher layer to back up, while the lower layer remains immune to congestion. Even a whole meshed subnetwork can be made immune to interior congestion by limiting ingress capacity and sufficient sizing of interior links, e.g. a non-blocking fat-tree network [Leiserson85]. An alternative to fat links near the root is numerous thin links with multi-path routing to ensure even worst-case patterns of load cannot congest any link, e.g. a Clos network [Clos53].

4. Feed-Forward-and-Up Mode: Guidelines for Adding Congestion Notification

Feed-forward-and-up is the mode already used for signalling ECN up the layers through MPLS into IP [RFC5129] and through IP-in-IP tunnels [RFC6040], whether encapsulating with IPv4 [RFC2003], IPv6 [RFC2473] or IPsec [RFC4301]. These RFCs take a consistent approach and the following guidelines are designed to ensure this consistency continues as ECN support is added to other protocols that encapsulate IP. The guidelines are also designed to ensure compliance with the

more general best current practice for the design of alternate ECN schemes given in [RFC4774] and extended by [RFC8311].

The rest of this section is structured as follows:

* Section 4.1 addresses the most straightforward cases, where [RFC6040] can be applied directly to add ECN to tunnels that are effectively IP-in-IP tunnels, but with shim header(s) between the IP headers.

* The subsequent sections give guidelines for adding ECN to a subnet technology that uses feed-forward-and-up mode like IP, but it is not so similar to IP that [RFC6040] rules can be applied directly. Specifically:

  - Sections 4.2, 4.3 and 4.4 respectively address how to add ECN support to the wire protocol and to the encapsulators and decapsulators at the ingress and egress of the subnet.

  - Section 4.5 deals with the special, but common, case of sequences of tunnels or subnets that all use the same technology

  - Section 4.6 deals with the question of reframing when IP packets do not map 1:1 into lower layer frames.

4.1. IP-in-IP Tunnels with Shim Headers

A common pattern for many tunnelling protocols is to encapsulate an inner IP header with shim header(s) then an outer IP header. A shim header is defined as one that is not sufficient alone to forward the packet as an outer header. Another common pattern is for a shim to encapsulate a layer 2 (L2) header, which in turn encapsulates (or might encapsulate) an IP header. [I-D.ietf-tsvwg-rfc6040update-shim] clarifies that RFC 6040 is just as applicable when there are shim(s) and possibly a L2 header between two IP headers.

However, it is not always feasible or necessary to propagate ECN between IP headers when separated by a shim. For instance, it might be too costly to dig to arbitrary depths to find an inner IP header, there may be little or no congestion within the tunnel by design (see null mode in Section 3.4 above), or a legacy implementation might not support ECN. In cases where a tunnel does not support ECN, it is important that the ingress does not copy the ECN field from an inner IP header to an outer. Therefore Section 4 of [I-D.ietf-tsvwg-rfc6040update-shim] requires network operators to configure the ingress of a tunnel that does not support ECN so that it zeros the ECN field in the outer IP header.

Nonetheless, in many cases it is feasible to propagate the ECN field between IP headers separated by shim header(s) and/or a L2 header. Particularly in the typical case when the outer IP header and the shim(s) are added (or removed) as part of the same procedure. Even if the shim(s) encapsulate a L2 header, it is often possible to find an inner IP header within the L2 PDU and propagate ECN between that and the outer IP header. This can be thought of as a special case of the feed-up-and-forward mode (Section 3.2), so the guidelines for this mode apply (Section 5).

Numerous shim protocols have been defined for IP tunnelling. More recent ones e.g. Geneve [RFC8926] and Generic UDP Encapsulation (GUE) [I-D.ietf-intarea-gue] cite and follow RFC 6040. And some earlier ones, e.g. CAPWAP [RFC5415] and LISP [RFC9300], cite RFC 3168, which is compatible with RFC 6040.

However, as Section 9.3 of [RFC3168] pointed out, ECN support needs to be defined for many earlier shim-based tunnelling protocols, e.g. L2TPv2 [RFC2661], L2TPv3 [RFC3931], GRE [RFC2784], PPTP [RFC2637], GTP [GTPv1], [GTPv1-U], [GTPv2-C] and Teredo [RFC4380] as well as some recent ones, e.g. VXLAN [RFC7348], NVGRE [RFC7637] and NSH [RFC8300].

All these IP-based encapsulations can be updated in one shot by simple reference to RFC 6040. However, it would not be appropriate to update all these protocols from within the present guidance document. Instead a companion specification [I-D.ietf-tsvwg-rfc6040update-shim] has been prepared that has the appropriate standards track status to update standards track protocols. For those that are not under IETF change control [I-D.ietf-tsvwg-rfc6040update-shim] can only recommend that the relevant body updates them.

4.2. Wire Protocol Design: Indication of ECN Support

This section is intended to guide the redesign of any lower layer protocol that encapsulate IP to add native ECN support at the lower layer. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

A lower layer (or subnet) congestion notification system:

1. SHOULD NOT apply explicit congestion notifications to PDUs that are destined for legacy layer-4 transport implementations that will not understand ECN, and

2. SHOULD NOT apply explicit congestion notifications to PDUs if the egress of the subnet might not propagate congestion notifications onward into the higher layer.

   We use the term ECN-PDUs for a PDU on a feedback loop that will propagate congestion notification properly because it meets both the above criteria. And a Not-ECN-PDU is a PDU on a feedback loop that does not meet at least one of the criteria, and will therefore not propagate congestion notification properly. A corollary of the above is that a lower layer congestion notification protocol:

3. SHOULD be able to distinguish ECN-PDUs from Not-ECN-PDUs.

Note that there is no need for all interior nodes within a subnet to be able to mark congestion explicitly. A mix of ECN and drop signals from different nodes is fine. However, if _any_ interior nodes might generate ECN markings, guideline 2 above says that all relevant egress node(s) SHOULD be able to propagate those markings up to the higher layer.

In IP, if the ECN field in each PDU is cleared to the Not-ECT (not ECN-capable transport) codepoint, it indicates that the L4 transport will not understand congestion markings. A congested buffer must not mark these Not-ECT PDUs, and therefore has to signal congestion by increasingly applying drop instead.

The mechanism a lower layer uses to distinguish the ECN-capability of PDUs need not mimic that of IP. The above guidelines merely say that the lower layer system, as a whole, should achieve the same outcome. For instance, ECN-capable feedback loops might use PDUs that are identified by a particular set of labels or tags. Alternatively, logical link protocols that use flow state might determine whether a PDU can be congestion marked by checking for ECN-support in the flow state. Other protocols might depend on out-of-band control signals.

The per-domain checking of ECN support in MPLS [RFC5129] is a good example of a way to avoid sending congestion markings to L4 transports that will not understand them, without using any header space in the subnet protocol.

In MPLS, header space is extremely limited, therefore RFC5129 does not provide a field in the MPLS header to indicate whether the PDU is an ECN-PDU or a Not-ECN-PDU. Instead, interior nodes in a domain are allowed to set explicit congestion indications without checking whether the PDU is destined for a L4 transport that will understand them. Nonetheless, this is made safe by requiring that the network operator upgrades all decapsulating edges of a whole domain at once,

as soon as even one switch within the domain is configured to mark rather than drop some PDUs during congestion. Therefore, any edge node that might decapsulate a packet will be capable of checking whether the higher layer transport is ECN-capable. When decapsulating a CE-marked packet, if the decapsulator discovers that the higher layer (inner header) indicates the transport is not ECN-capable, it drops the packet effectively on behalf of the earlier congested node (see Decapsulation Guideline 1 in Section 4.4).

It was only appropriate to define such an incremental deployment strategy because MPLS is targeted solely at professional operators, who can be expected to ensure that a whole subnetwork is consistently configured. This strategy might not be appropriate for other link technologies targeted at zero-configuration deployment or deployment by the general public (e.g. Ethernet). For such 'plug-and-play' environments it will be necessary to invent a failsafe approach that ensures congestion markings will never fall into black holes, no matter how inconsistently a system is put together. Alternatively, congestion notification relying on correct system configuration could be confined to flavours of Ethernet intended only for professional network operators, such as Provider Backbone Bridges (PBB [IEEE802.1Q]; previously 802.1ah).

ECN support in TRILL [I-D.ietf-trill-ecn-support] provides a good example of how to add ECN to a lower layer protocol without relying on careful and consistent operator configuration. TRILL provides an extension header word with space for flags of different categories depending on whether logic to understand the extension is critical. The congestion experienced marking has been defined as a 'critical ingress-to-egress' flag. So if a transit RBridge sets this flag on a frame and an egress RBridge does not have any logic to process it, it will drop it; which is the desired default action anyway. Therefore TRILL RBridges can be updated with support for ECN in no particular order and, at the egress of the TRILL campus, congestion notification will be propagated to IP as ECN whenever ECN logic has been implemented at the egress, or as drop otherwise.

QCN [IEEE802.1Q] is not intended to extend beyond a single subnet, or to interoperate with ECN. Nonetheless, the way QCN indicates to lower layer devices that the end-points will not understand QCN provides another example that a lower layer protocol designer might be able to mimic for their scenario. An operator can define certain Priority Code Points (PCPs [IEEE802.1Q]; previously 802.1p) to indicate non-QCN frames and an ingress bridge is required to map arriving not-QCN-capable IP packets to one of these non-QCN PCPs.

When drop for non-ECN traffic is deferred to the egress of a subnet, it cannot necessarily be assumed that one ECN mark is equivalent to one drop, as was originally required by [RFC3168]. [RFC8311] updated RFC 3168, to allow experimentation with congestion markings that are not equivalent to drop, in particular for L4S [RFC9331]. ECN support in TRILL [I-D.ietf-trill-ecn-support] is a good example of a way to defer drop to the egress of a subnet both when marks are equivalent to drops (as in RFC 3168) and when they are not (as in L4S). The ECN scheme for MPLS [RFC5129] was defined before L4S, so it only currently supports deferred drop that is equivalent to ECN-marking. Nonetheless, in principle, MPLS (and potentially future L2 protocols) could support L4S marking and copy TRILL's approach for determining the drop level of any non-ECN traffic at the subnet egress.

4.3. Encapsulation Guidelines

This section is intended to guide the redesign of any node that encapsulates IP with a lower layer header when adding native ECN support to the lower layer protocol. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

1. Egress Capability Check: A subnet ingress needs to be sure that the corresponding egress of a subnet will propagate any congestion notification added to the outer header across the subnet. This is necessary in addition to checking that an incoming PDU indicates an ECN-capable (L4) transport. Examples of how this guarantee might be provided include:

   * by configuration (e.g. if any label switches in a domain support ECN marking, [RFC5129] requires all egress nodes to have been configured to propagate ECN)

   * by the ingress explicitly checking that the egress propagates ECN (e.g. an early attempt to add ECN support to TRILL used IS-IS to check path capabilities before adding ECN extension flags to each frame [RFC7780]).

   * by inherent design of the protocol (e.g. by encoding ECN marking on the outer header in such a way that a legacy egress that does not understand ECN will consider the PDU corrupt or invalid and discard it, thus at least propagating a form of congestion signal).

2. Egress Fails Capability Check: If the ingress cannot guarantee that the egress will propagate congestion notification, the ingress SHOULD disable ECN at the lower layer when it forwards

the PDU. An example of how the ingress might disable ECN at the lower layer would be by setting the outer header of the PDU to identify it as a Not-ECN-PDU, assuming the subnet technology supports such a concept.

3. Standard Congestion Monitoring Baseline: Once the ingress to a subnet has established that the egress will correctly propagate ECN, on encapsulation it SHOULD encode the same level of congestion in outer headers as is arriving in incoming headers. For example, it might copy any incoming congestion notification into the outer header of the lower layer protocol.

   This ensures that bulk congestion monitoring of outer headers (e.g. by a network management node monitoring ECN in passing frames) will measure congestion accumulated along the whole upstream path starting from the Load Regulator not just starting from the ingress of the subnet. A node that is not the Load Regulator SHOULD NOT re-initialize the level of CE markings in the outer to zero.

   It would still also be possible to measure congestion introduced across one subnet (or tunnel) by subtracting the level of CE markings on inner headers from that on outer headers (see Appendix C of [RFC6040]). For example:

   * If this guideline has been followed and if the level of CE markings is 0.4% on the outer and 0.1% on the inner, 0.4% congestion has been introduced across all the networks since the load regulator, and 0.3% (= 0.4% - 0.1%) has been introduced since the ingress to the current subnet (or tunnel);

   * Without this guideline, if the subnet ingress had re-initialized the outer congestion level to zero, the outer and inner would measure 0.1% and 0.3%. It would still be possible to infer that the congestion introduced since the Load Regulator was 0.4% (= 0.1% + 0.3%). But only if the monitoring system somehow knows whether the subnet ingress re-initialized the congestion level.

   As long as subnet and tunnel technologies use the standard congestion monitoring baseline in this guideline, monitoring systems will know to use the former approach, rather than having to "somehow know" which approach to use.

4.4. Decapsulation Guidelines

This section is intended to guide the redesign of any node that decapsulates IP from within a lower layer header when adding native ECN support to the lower layer protocol. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

A subnet egress SHOULD NOT simply copy congestion notification from outer headers to the forwarded header. It SHOULD calculate the outgoing congestion notification field from the inner and outer headers using the following guidelines. If there is any conflict, rules earlier in the list take precedence over rules later in the list:

1. If the arriving inner header is a Not-ECN-PDU it implies the L4 transport will not understand explicit congestion markings. Then:

   * If the outer header carries an explicit congestion marking, it is likely that a protocol error has occurred, so drop is the only indication of congestion that the L4 transport will understand. If the congestion marking is the most severe possible, the packet MUST be dropped. However, if congestion can be marked with multiple levels of severity and the packet's marking is not the most severe, this requirement can be relaxed to: the packet SHOULD be dropped.

   * If the outer is an ECN-PDU that carries no indication of congestion or a Not-ECN-PDU the PDU SHOULD be forwarded, but still as a Not-ECN-PDU.

2. If the outer header does not support explicit congestion notification (a Not-ECN-PDU), but the inner header does (an ECN-PDU), the inner header SHOULD be forwarded unchanged.

3. In some lower layer protocols congestion may be signalled as a numerical level, such as in the control frames of quantized congestion notification (QCN [IEEE802.1Q]). If such a multi-bit encoding encapsulates an ECN-capable IP data packet, a function will be needed to convert the quantized congestion level into the frequency of congestion markings in outgoing IP packets.

4. Congestion indications might be encoded by a severity level. For instance increasing levels of congestion might be encoded by numerically increasing indications, e.g. pre-congestion notification (PCN) can be encoded in each PDU at three severity

levels in IP or MPLS [RFC6660] and the default encapsulation and decapsulation rules [RFC6040] are compatible with this interpretation of the ECN field.

If the arriving inner header is an ECN-PDU, where the inner and outer headers carry indications of congestion of different severity, the more severe indication SHOULD be forwarded in preference to the less severe.

5. The inner and outer headers might carry a combination of congestion notification fields that should not be possible given any currently used protocol transitions. For instance, if Encapsulation Guideline 3 in Section 4.3 had been followed, it should not be possible to have a less severe indication of congestion in the outer than in the inner. It MAY be appropriate to log unexpected combinations of headers and possibly raise an alarm.

   If a safe outgoing codepoint can be defined for such a PDU, the PDU SHOULD be forwarded rather than dropped. Some implementers discard PDUs with currently unused combinations of headers just in case they represent an attack. However, an approach using alarms and policy-mediated drop is preferable to hard-coded drop, so that operators can keep track of possible attacks but currently unused combinations are not precluded from future use through new standards actions.

4.5. Sequences of Similar Tunnels or Subnets

In some deployments, particularly in 3GPP networks, an IP packet may traverse two or more IP-in-IP tunnels in sequence that all use identical technology (e.g. GTP).

In such cases, it would be sufficient for every encapsulation and decapsulation in the chain to comply with RFC 6040. Alternatively, as an optimisation, a node that decapsulates a packet and immediately re-encapsulates it for the next tunnel MAY copy the incoming outer ECN field directly to the outgoing outer and the incoming inner ECN field directly to the outgoing inner. Then the overall behavior across the sequence of tunnel segments would still be consistent with RFC 6040.

Appendix C of RFC6040 describes how a tunnel egress can monitor how much congestion has been introduced within a tunnel. A network operator might want to monitor how much congestion had been introduced within a whole sequence of tunnels. Using the technique in Appendix C of RFC6040 at the final egress, the operator could monitor the whole sequence of tunnels, but only if the above

optimisation were used consistently along the sequence of tunnels, in order to make it appear as a single tunnel. Therefore, tunnel endpoint implementations SHOULD allow the operator to configure whether this optimisation is enabled.

When ECN support is added to a subnet technology, consideration SHOULD be given to a similar optimisation between subnets in sequence if they all use the same technology.

4.6. Reframing and Congestion Markings

The guidance in this section is worded in terms of framing boundaries, but it applies equally whether the protocol data units are frames, cells or packets.

Where an AQM marks the ECN field of IP packets as they queue into a layer-2 link, there will be no problem with framing boundaries, because the ECN markings would be applied directly to IP packets. The guidance in this section is only applicable where an ECN capability is being added to a layer-2 protocol so that layer-2 frames can be ECN-marked by an AQM at layer-2. This would only be necessary where AQM will be applied at pure layer-2 nodes (without IP-awareness).

Where ECN marking has had to be applied at non-IP-aware nodes and framing boundaries do not necessarily align with packet boundaries, the decapsulating IP forwarding node SHOULD propagate ECN markings from layer-2 frame headers to IP packets that may have different boundaries as a consequence of reframing.

Two possible design goals for propagating congestion indications, described in Section 5.3 of [RFC3168] and Section 2.4 of [RFC7141], are:

1. approximate preservation of the presence (and therefore timing) of congestion marks on the L2 frames used to construct an IP packet;

    a. at high frequency of congestion marking, approximate preservation of the proportion of congestion marks arriving and departing;

    b. at low frequency of congestion marking, approximate preservation of the timing of congestion marks arriving and departing.

In either case, an implementation SHOULD ensure that any new incoming congestion indication is propagated immediately, not held awaiting the possibility of further congestion indications to be sufficient to indicate congestion on an outgoing PDU [RFC7141]. Nonetheless, to facilitate pipelined implementation, it would be acceptable for congestion marks to propagate to a slightly later IP packet.

At decapsulation in either case:

* ECN marking propagation logically occurs before application of Decapsulation Guideline 1 in Section 4.4. For instance, if ECN marking propagation would cause an ECN congestion indication to be applied to an IP packet that is a Not-ECN-PDU, then that IP packet is dropped in accordance with Guideline 1;

* where a mix of ECN-PDUs and non-ECN-PDUs arrives to construct the same IP packet, the decapsulation spec SHOULD require that packet to be discarded.

* where a mix of different types of ECN-PDUs arrives to construct the same IP packet, e.g. a mix of frames that map to ECT(0) and ECT(1) IP packets, the decapsulation spec might consider this a protocol error. But, if the lower layer protocol has defined such a mix of types of ECN-PDU as valid, it SHOULD require the resulting IP packet to be set to either ECT(0) or ECT(1). In this case, it SHOULD take into account that the RFC series has so far allowed ECT(0) and ECT(1) to be considered equivalent [RFC3168], or ECT(1) can provide a less severe congestion marking than CE [RFC6040], or ECT(1) can indicate an unmarked but ECN-capable packet that is subject to a different marking algorithm to ECT(0) packets, for example L4S [RFC8311] [RFC9331].

The following are two ways that goal 1 might be achieved, but they are not intended to be the only ways:

* Every IP PDU that is constructed, in whole or in part, from an L2 frame that is marked with a congestion signal, has that signal propagated to it;

* Every L2 frame that is marked with a congestion signal, propagates that signal to one IP PDU which is constructed, in whole or in part, from it. If multiple IP PDUs meet this description, the choice can be made arbitrarily but ought to be consistent.

The following gives one way that goal 2 might be achieved, but it is not intended to be the only way:

* For each of the streams of frames that encapsulate the IP packets of each IP-ECN codepoint and follow the same path through the subnet, a counter ('in') tracks octets arriving within the payload of marked L2 frames and another ('out') tracks octets departing in marked IP packets. While 'in' exceeds 'out', forwarded IP packets are ECN-marked. If 'out' exceeds 'in' for longer than a timeout, both counters are zeroed, to ensure that the start of the next congestion episode propagates immediately. The 'out' counter includes octets in reconstructed IP packets that would have been marked, but had to be dropped because they were Not-ECN-PDUs (by Decapsulation Guideline 1 in Section 4.4).

Generally, the number of L2 frames may be higher (e.g. ATM), similar to, or lower (e.g. 802.11 aggregation at a L2-only station) than the number of IP PDUs, and this distinction may influence the choice of mechanism.

5. Feed-Up-and-Forward Mode: Guidelines for Adding Congestion Notification

The guidance in this section is applicable, for example, when IP packets:

* are encapsulated in Ethernet headers, which have no support for ECN;

* are forwarded by the eNode-B (base station) of a 3GPP radio access network, which is required to apply ECN marking during congestion, [LTE-RA], [UTRAN], but the Packet Data Convergence Protocol (PDCP) that encapsulates the IP header over the radio access has no support for ECN.

This guidance also generalizes to encapsulation by other subnet technologies with no native support for explicit congestion notification at the lower layer, but with support for finding and processing an IP header. It is unlikely to be applicable or necessary for IP-in-IP encapsulation, where feed-forward-and-up mode based on [RFC6040] would be more appropriate.

Marking the IP header while switching at layer-2 (by using a layer-3 switch) or while forwarding in a radio access network seems to represent a layering violation. However, it can be considered as a benign optimisation if the guidelines below are followed. Feed-up-and-forward is certainly not a general alternative to implementing feed-forward congestion notification in the lower layer, because:

* IPv4 and IPv6 are not the only layer-3 protocols that might be encapsulated by lower layer protocols

* Link-layer encryption might be in use, making the layer-2 payload inaccessible

* Many Ethernet switches do not have 'layer-3 switch' capabilities so they cannot read or modify an IP payload

* It might be costly to find an IP header (IPv4 or IPv6) when it may be encapsulated by more than one lower layer header, e.g. Ethernet MAC in MAC ([IEEE802.1Q]; previously 802.1ah).

Nonetheless, configuring lower layer equipment to look for an ECN field in an encapsulated IP header is a useful optimisation. If the implementation follows the guidelines below, this optimisation does not have to be confined to a controlled environment such as within a data centre; it could usefully be applied on any network even if the operator is not sure whether the above issues will never apply:

1. If a native lower-layer congestion notification mechanism exists for a subnet technology, it is safe to mix feed-up-and-forward with feed-forward-and-up on other switches in the same subnet. However, it will generally be more efficient to use the native mechanism.

2. The depth of the search for an IP header SHOULD be limited. If an IP header is not found soon enough, or an unrecognized or unreadable header is encountered, the switch SHOULD resort to an alternative means of signalling congestion (e.g. drop, or the native lower layer mechanism if available).

3. It is sufficient to use the first IP header found in the stack; the egress of the relevant tunnel can propagate congestion notification upwards to any more deeply encapsulated IP headers later.

6. Feed-Backward Mode: Guidelines for Adding Congestion Notification

It can be seen from Section 3.3 that congestion notification in a subnet using feed-backward mode has generally not been designed to be directly coupled with IP layer congestion notification. The subnet attempts to minimize congestion internally, and if the incoming load at the ingress exceeds the capacity somewhere through the subnet, the layer 3 buffer into the ingress backs up. Thus, a feed-backward mode subnet is in some sense similar to a null mode subnet, in that there is no need for any direct interaction between the subnet and higher layer congestion notification. Therefore no detailed protocol design guidelines are appropriate. Nonetheless, a more general guideline is appropriate:

> A subnetwork technology intended to eventually interface to IP SHOULD NOT be designed using only the feed-backward mode, which is certainly best for a stand-alone subnet, but would need to be modified to work efficiently as part of the wider Internet, because IP uses feed-forward-and-up mode.

The feed-backward approach at least works beneath IP, where the term 'works' is used only in a narrow functional sense because feed-backward can result in very inefficient and sluggish congestion control except if it is confined to the subnet directly connected to the original data source, when it is faster than feed-forward. It would be valid to design a protocol that could work in feed-backward mode for paths that only cross one subnet, and in feed-forward-and-up mode for paths that cross subnets.

In the early days of TCP/IP, a similar feed-backward approach was tried for explicit congestion signalling, using source-quench (SQ) ICMP control packets. However, SQ fell out of favour and is now formally deprecated [RFC6633]. The main problem was that it is hard for a data source to tell the difference between a spoofed SQ message and a quench request from a genuine buffer on the path. It is also hard for a lower layer buffer to address an SQ message to the original source port number, which may be buried within many layers of headers, and possibly encrypted.

QCN (also known as backward congestion notification, BCN; see Sections 3033 of [IEEE802.1Q]; previously known as 802.1Qau) uses a feed-backward mode structurally similar to ATM's relative rate mechanism. However, QCN confines its applicability to scenarios such as some data centres where all endpoints are directly attached by the same Ethernet technology. If a QCN subnet were later connected into a wider IP-based internetwork (e.g. when attempting to interconnect multiple data centres) it would suffer the inefficiency shown in Figure 3.

7. IANA Considerations

This section is to be removed before publishing as an RFC.

This memo includes no request to IANA.

8. Security Considerations

If a lower layer wire protocol is redesigned to include explicit congestion signalling in-band in the protocol header, care SHOULD be taken to ensure that the field used is specified as mutable during transit. Otherwise interior nodes signalling congestion would invalidate any authentication protocol applied to the lower layer header by altering a header field that had been assumed as immutable.

The redesign of protocols that encapsulate IP in order to propagate congestion signals between layers raises potential signal integrity concerns. Experimental or proposed approaches exist for assuring the end-to-end integrity of in-band congestion signals, e.g.:

* Congestion exposure (ConEx) for networks to audit that their congestion signals are not being suppressed by other networks or by receivers, and for networks to police that senders are responding sufficiently to the signals, irrespective of the L4 transport protocol used [RFC7713].

* A test for a sender to detect whether a network or the receiver is suppressing congestion signals (for example see 2nd para of Section 20.2 of [RFC3168]).

Given these end-to-end approaches are already being specified, it would make little sense to attempt to design hop-by-hop congestion signal integrity into a new lower layer protocol, because end-to-end integrity inherently achieves hop-by-hop integrity.

Section 6 gives vulnerability to spoofing as one of the reasons for deprecating feed-backward mode.

9. Conclusions

Following the guidance in this document enables ECN support to be extended consistently to numerous protocols that encapsulate IP (IPv4 and IPv6), so that IP continues to fulfil its role as an end-to-end interoperability layer. This includes:

* A wide range of tunnelling protocols including those with various forms of shim header between two IP headers, possibly also separated by a L2 header;

* A wide range of subnet technologies, particularly those that work in the same 'feed-forward-and-up' mode that is used to support ECN in IP and MPLS.

Guidelines have been defined for supporting propagation of ECN between Ethernet and IP on so-called Layer-3 Ethernet switches, using a 'feed-up-and-forward' mode. This approach could enable other subnet technologies to pass ECN signals into the IP layer, even if they do not support ECN natively.

Finally, attempting to add ECN to a subnet technology in feed-backward mode is deprecated except in special cases, due to its likely sluggish response to congestion.

10. References

10.1. Normative References

[I-D.ietf-trill-ecn-support]
Eastlake, D. E. and B. Briscoe, "TRILL (TRansparent Interconnection of Lots of Links): ECN (Explicit Congestion Notification) Support", Work in Progress, Internet-Draft, draft-ietf-trill-ecn-support-07, 25 February 2018, <https://datatracker.ietf.org/doc/html/draft-ietf-trill-ecn-support-07>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <https://www.rfc-editor.org/info/rfc3819>.

[RFC4774] Floyd, S., "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field", BCP 124, RFC 4774, DOI 10.17487/RFC4774, November 2006, <https://www.rfc-editor.org/info/rfc4774>.

[RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <https://www.rfc-editor.org/info/rfc5129>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <https://www.rfc-editor.org/info/rfc6040>.

[RFC7141] Briscoe, B. and J. Manner, "Byte and Packet Congestion Notification", BCP 41, RFC 7141, DOI 10.17487/RFC7141, February 2014, <https://www.rfc-editor.org/info/rfc7141>.

10.2. Informative References

[ATM-TM-ABR]
Cisco, "Understanding the Available Bit Rate (ABR) Service Category for ATM VCs", Design Technote 10415, 5 June 2005, <https://www.cisco.com/c/en/us/support/docs/asynchronous-transfer-mode-atm/atm-traffic-management/10415-atmabr.html>.

[Buck00] Buckwalter, J.T., "Frame Relay: Technology and Practice", Pub. Addison Wesley ISBN-13: 978-0201485240, 2000.

[Clos53] Clos, C., "A Study of Non-Blocking Switching Networks", Bell Systems Technical Journal 32(2):406424, March 1953.

[GTPv1] 3GPP, "GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface", Technical Specification TS 29.060.

[GTPv1-U] 3GPP, "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", Technical Specification TS 29.281.

[GTPv2-C] 3GPP, "Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C)", Technical Specification TS 29.274.

[I-D.ietf-intarea-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", Work in Progress, Internet-Draft, draft-ietf-intarea-gue-09, 26 October 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-intarea-gue-09>.

[I-D.ietf-tsvwg-rfc6040update-shim]
Briscoe, B., "Propagating Explicit Congestion Notification Across IP Tunnel Headers Separated by a Shim", Work in Progress, Internet-Draft, draft-ietf-tsvwg-rfc6040update-shim-22, 29 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-rfc6040update-shim-22>.

[IEEE802.1Q]
IEEE, "IEEE Standard for Local and Metropolitan Area NetworksVirtual Bridged Local Area NetworksAmendment 6: Provider Backbone Bridges", IEEE Std 802.1Q-2018, July 2018, <https://ieeexplore.ieee.org/document/8403927>.

[ITU-T.I.371]
ITU-T, "Traffic Control and Congestion Control in B-ISDN", ITU-T Rec. I.371 (03/04), March 2004, <https://www.itu.int/rec/T-REC-I.371>.

[Leiserson85]
Leiserson, C.E., "Fat-trees: universal networks for hardware-efficient supercomputing", IEEE Transactions on Computers 34(10):892901, October 1985.

[LTE-RA] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2", Technical Specification TS 36.300.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <https://www.rfc-editor.org/info/rfc2003>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

[RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, DOI 10.17487/RFC2637, July 1999, <https://www.rfc-editor.org/info/rfc2637>.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, DOI 10.17487/RFC2661, August 1999, <https://www.rfc-editor.org/info/rfc2661>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <https://www.rfc-editor.org/info/rfc2784>.

[RFC2884] Hadi Salim, J. and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", RFC 2884, DOI 10.17487/RFC2884, July 2000, <https://www.rfc-editor.org/info/rfc2884>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <https://www.rfc-editor.org/info/rfc2983>.

[RFC3931] Lau, J., Ed., Townsley, M., Ed., and I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, DOI 10.17487/RFC3931, March 2005, <https://www.rfc-editor.org/info/rfc3931>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <https://www.rfc-editor.org/info/rfc4380>.

[RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <https://www.rfc-editor.org/info/rfc5415>.

[RFC6633] Gont, F., "Deprecation of ICMP Source Quench Messages", RFC 6633, DOI 10.17487/RFC6633, May 2012, <https://www.rfc-editor.org/info/rfc6633>.

[RFC6660] Briscoe, B., Moncaster, T., and M. Menth, "Encoding Three Pre-Congestion Notification (PCN) States in the IP Header Using a Single Diffserv Codepoint (DSCP)", RFC 6660, DOI 10.17487/RFC6660, July 2012, <https://www.rfc-editor.org/info/rfc6660>.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <https://www.rfc-editor.org/info/rfc7323>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/info/rfc7348>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <https://www.rfc-editor.org/info/rfc7567>.

[RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <https://www.rfc-editor.org/info/rfc7637>.

[RFC7713] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements", RFC 7713, DOI 10.17487/RFC7713, December 2015, <https://www.rfc-editor.org/info/rfc7713>.

[RFC7780] Eastlake 3rd, D., Zhang, M., Perlman, R., Banerjee, A., Ghanwani, A., and S. Gupta, "Transparent Interconnection of Lots of Links (TRILL): Clarifications, Corrections, and Updates", RFC 7780, DOI 10.17487/RFC7780, February 2016, <https://www.rfc-editor.org/info/rfc7780>.

[RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <https://www.rfc-editor.org/info/rfc8084>.

[RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <https://www.rfc-editor.org/info/rfc8087>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8257] Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L., and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers", RFC 8257, DOI 10.17487/RFC8257, October 2017, <https://www.rfc-editor.org/info/rfc8257>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

[RFC8311] Black, D., "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation", RFC 8311, DOI 10.17487/RFC8311, January 2018, <https://www.rfc-editor.org/info/rfc8311>.

[RFC8926] Gross, J., Ed., Ganga, I., Ed., and T. Sridhar, Ed., "Geneve: Generic Network Virtualization Encapsulation", RFC 8926, DOI 10.17487/RFC8926, November 2020, <https://www.rfc-editor.org/info/rfc8926>.

[RFC9300] Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos, Ed., "The Locator/ID Separation Protocol (LISP)", RFC 9300, DOI 10.17487/RFC9300, October 2022, <https://www.rfc-editor.org/info/rfc9300>.

[RFC9331] De Schepper, K. and B. Briscoe, Ed., "The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S)", RFC 9331, DOI 10.17487/RFC9331, January 2023, <https://www.rfc-editor.org/info/rfc9331>.

[UTRAN] 3GPP, "UTRAN Overall Description", Technical Specification TS 25.401.

Comments Solicited

This section is to be removed before publishing as an RFC.

Comments and questions are encouraged and very welcome. They can be addressed to the IETF Transport Area working group mailing list <tsvwg@ietf.org>, and/or to the authors.

Acknowledgements

Thanks to Gorry Fairhurst and David Black for extensive reviews. Thanks also to the following reviewers: Joe Touch, Andrew McGregor, Richard Scheffenegger, Ingemar Johansson, Piers O'Hanlon, Donald Eastlake, Jonathan Morton, Markku Kojo, Sebastian Möller, Martin Duke and Michael Welzl, who pointed out that lower layer congestion notification signals may have different semantics to those in IP. Thanks are also due to the tsvwg chairs, TSV ADs and IETF liaison people such as Eric Gray, Dan Romascanu and Gonzalo Camarillo for helping with the liaisons with the IEEE and 3GPP. And thanks to Georg Mayer and particularly to Erik Guttman for the extensive search and categorisation of any 3GPP specifications that cite ECN specifications. Thanks also to the Area Reviewers Dan Harkins, Paul Kyzivat, Sue Hares and Dale Worley.

Bob Briscoe was part-funded by the European Community under its Seventh Framework Programme through the Trilogy project (ICT-216372) for initial drafts then through the Reducing Internet Transport Latency (RITE) project (ICT-317700), and for final drafts (from -18) he was funded by Apple Inc. The views expressed here are solely those of the authors.

Contributors

Pat Thaler
Broadcom Corporation (retired)
CA
USA

Pat was a co-author of this draft, but retired before its publication.

Authors' Addresses

Bob Briscoe
Independent
United Kingdom
Email: ietf@bobbriscoe.net
URI: https://bobbriscoe.net/

John Kaippallimalil
Futurewei
5700 Tennyson Parkway, Suite 600
Plano, Texas 75024
United States of America
Email: kjohn@futurewei.com

Network Working Group
Internet-Draft
Intended status: Standard Track

Lucy Yong(Ed.)
Huawei Technologies
E. Crabbe
Oracle
X. Xu
Huawei Technologies
T. Herbert
Facebook

Expires: February 2017 September 30, 2016

GRE-in-UDP Encapsulation
draft-ietf-tsvwg-gre-in-udp-encap-19

Abstract

This document specifies a method of encapsulating network protocol packet within GRE and UDP headers. This GRE-in-UDP encapsulation allows the UDP source port field to be used as an entropy field. This may be used for load balancing of GRE traffic in transit networks using existing ECMP mechanisms. There are two applicability scenarios for GRE-in-UDP with different requirements: (1) general Internet; (2) a traffic-managed controlled environment. The controlled environment has less restrictive requirements than the general Internet.

Status of This Document

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 30,2017.

Table of Contents

1. Introduction

This document specifies a generic GRE-in-UDP encapsulation for tunneling network protocol packets across an IP network based on Generic Routing Encapsulation (GRE) [RFC2784][RFC7676] and User Datagram Protocol(UDP) [RFC768] headers. The GRE header indicates the payload protocol type via an EtherType [RFC7042] in the protocol type field, and the source port field in the UDP header may be used to provide additional entropy.

A GRE-in-UDP tunnel offers the possibility of better performance for load balancing GRE traffic in transit networks using existing Equal-Cost Multi-Path (ECMP) mechanisms that use a hash of the five-tuple of source IP address, destination IP address, UDP/TCP source port, UDP/TCP destination port. While such hashing distributes UDP and Transmission Control Protocol (TCP)[RFC793] traffic between a common pair of IP addresses across paths, it uses a single path for corresponding GRE traffic because only the two IP addresses and protocol/next header fields participate in the ECMP hash. Encapsulating GRE in UDP enables use of the UDP source port to provide entropy to ECMP hashing.

In addition, GRE-in-UDP enables extending use of GRE across networks that otherwise disallow it; for example, GRE-in-UDP may be used to bridge two islands where GRE is not supported natively across the middleboxes.

GRE-in-UDP encapsulation may be used to encapsulate already tunneled traffic, i.e., tunnel-in-tunnel. In this case, GRE-in-UDP tunnels treat the endpoints of the outer tunnel as the end hosts; the presence of an inner tunnel does not change the outer tunnel’s handling of network traffic.

A GRE-in-UDP tunnel is capable of carrying arbitrary traffic and behaves as a UDP application on an IP network. However, a GRE-in-UDP tunnel carrying certain types of traffic does not satisfy the requirements for UDP applications on the Internet [RFC5405bis]. GRE-in-UDP tunnels that do not satisfy these requirements MUST NOT be deployed to carry such traffic over the Internet. For this reason, this document specifies two deployment scenarios for GRE-in-UDP tunnels with GRE-in-UDP tunnel requirements for each of them: (1) general Internet; (2) a traffic-managed controlled environment (TMCE). The TMCE scenario has less restrictive technical requirements for the protocol but more restrictive management and operation requirements for the network by comparison to the general Internet scenario.

To provide security for traffic carried by a GRE-in-UDP tunnel, this document also specifies Datagram Transport Layer Security (DTLS) for GRE-in-UDP tunnels, which SHOULD be used when security is a concern.

GRE-in-UDP encapsulation usage requires no changes to the transit IP network. ECMP hash functions in most existing IP routers may utilize and benefit from the additional entropy enabled by GRE-in-UDP tunnels without any change or upgrade to their ECMP implementation. The encapsulation mechanism is applicable to a variety of IP networks including Data Center and Wide Area Networks, as well as both IPv4 and IPv6 networks.

1.1. Terminology

The terms defined in [RFC768] and [RFC2784] are used in this document. Following are additional terms used in this draft.

Decapsulator: a component performing packet decapsulation at tunnel egress.

ECMP: Equal-Cost Multi-Path.

Encapsulator: a component performing packet encapsulation at tunnel egress.

Flow Entropy: The information to be derived from traffic or applications and to be used by network devices in ECMP process [RFC6438].

Default GRE-in-UDP Tunnel: A GRE-in-UDP tunnel that can apply to the general Internet.

TMCE: A Traffic-managed controlled environment, i.e. an IP network that is traffic-engineered and/or otherwise managed (e.g., via use of traffic rate limiters) to avoid congestion, as defined in Section 2.

TMCE GRE-in-UDP Tunnel: A GRE-in-UDP tunnel that can only apply to a traffic-managed controlled environment that is defined in Section 2.

Tunnel Egress: A tunnel end point that performs packet decapsulation.

Tunnel Ingress: A tunnel end point that performs packet encapsulation.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Applicability Statement

GRE-in-UDP encapsulation applies to IPv4 and IPv6 networks; in both cases, encapsulated packets are treated as UDP datagrams. Therefore, a GRE-in-UDP tunnel needs to meet the UDP usage requirements specified in [RFC5405bis]. These requirements depend on both the delivery network and the nature of the encapsulated traffic. For example, the GRE-in-UDP tunnel protocol does not provide any congestion control functionality beyond that of the encapsulated traffic. Therefore, a GRE-in-UDP tunnel MUST be used only with congestion controlled traffic (e.g., IP unicast traffic) and/or within a network that is traffic-managed to avoid congestion.

[RFC5405bis] describes two applicability scenarios for UDP applications: 1) General Internet and 2) A controlled environment. The controlled environment means a single administrative domain or bilaterally agreed connection between domains. A network forming a controlled environment can be managed/operated to meet certain conditions while the general Internet cannot be; thus the requirements for a tunnel protocol operating under a controlled environment can be less restrictive than the requirements in the general Internet.

For the purpose of this document, a traffic-managed controlled environment (TMCE) is defined as an IP network that is traffic-engineered and/or otherwise managed (e.g., via use of traffic rate limiters) to avoid congestion.

This document specifies GRE-in-UDP tunnel usage in the general Internet and GRE-in-UDP tunnel usage in a traffic-managed controlled environment and uses "default GRE-in-UDP tunnel" and "TMCE GRE-in-UDP tunnel" terms to refer to each usage.

NOTE: Although this document specifies two different sets of GRE-in-UDP tunnel requirements based on tunnel usage, the tunnel implementation itself has no ability to detect how and where it is deployed. Therefore it is the responsibility of the user or operator who deploys a GRE-in-UDP tunnel to ensure that it meets the appropriate requirements.

2.1. GRE-in-UDP Tunnel Requirements

This section states out the requirements for a GRE-in-UDP tunnel. Section 2.1.1 describes the requirements for a default GRE-in-UDP tunnel that is suitable for the general Internet; Section 2.1.2 describes a set of relaxed requirements for a TMCE GRE-in-UDP tunnel used in a traffic-managed controlled environment. Both Sections 2.1.1 and 2.1.2 are applicable to an IPv4 or IPv6 delivery network.

2.1.1. Requirements for Default GRE-in-UDP Tunnel

The following is a summary of the default GRE-in-UDP tunnel requirements:

1. A UDP checksum SHOULD be used when encapsulating in IPv4.

2. A UDP checksum MUST be used when encapsulating in IPv6.

3. GRE-in-UDP tunnel MUST NOT be deployed or configured to carry traffic that is not congestion controlled. As stated in [RFC5405bis], IP-based unicast traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. A default GRE-in-UDP tunnel is not appropriate for traffic that is not known to be congestion-controlled (e.g., most IP multicast traffic).

4. UDP source port values that are used as a source of flow entropy SHOULD be chosen from the ephemeral port range (49152-65535) [RFC5405bis].

5. The use of the UDP source port MUST be configurable so that a single value can be set for all traffic within the tunnel (this disables use of the UDP source port to provide flow entropy). When a single value is set, a random port SHOULD be selected in order to minimize the vulnerability to off-path attacks [RFC6056].

6. For IPv6 delivery networks, the flow entropy SHOULD also be placed in the flow label field for ECMP per [RFC6438].

7. At the tunnel ingress, any fragmentation of the incoming packet (e.g., because the tunnel has a Maximum Transmission Unit (MTU) that is smaller than the packet) SHOULD be performed before encapsulation. In addition, the tunnel ingress MUST apply the UDP checksum to all encapsulated fragments so that the tunnel egress can validate reassembly of the fragments; it MUST set the same Differentiated Services Code Point (DSCP) value as in the Differentiated Services

(DS) field of the payload packet in all fragments [RFC2474]. To avoid unwanted forwarding over multiple paths, the same source UDP port value SHOULD be set in all packet fragments.

2.1.2. Requirements for TMCE GRE-in-UDP Tunnel

The section contains the TMCE GRE-in-UDP tunnel requirements. It lists the changed requirements, compared with a Default GRE-in-UDP Tunnel, for a TMCE GRE-in-UDP Tunnel, which corresponds to the requirements 1-3 listed in Section 2.1.1.

1. A UDP checksum SHOULD be used when encapsulating in IPv4. A tunnel endpoint sending GRE-in-UDP MAY disable the UDP checksum, since GRE has been designed to work without a UDP checksum [RFC2784]. However, a checksum also offers protection from mis-delivery to another port.

2. Use of UDP checksum MUST be the default when encapsulating in IPv6. This default MAY be overridden via configuration of UDP zero-checksum mode. All usage of UDP zero-checksum mode with IPv6 is subject to the additional requirements specified in Section 6.2.

3. A GRE-in-UDP tunnel MAY encapsulate traffic that is not congestion controlled.

The requirements 4-7 listed in Section 2.1.1 also apply to a TMCE GRE-in-UDP Tunnel.

3. GRE-in-UDP Encapsulation

The GRE-in-UDP encapsulation format contains a UDP header [RFC768] and a GRE header [RFC2890]. The format is shown as follows: (presented in bit order)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

 IPv4 Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Version|  IHL  |Type of Service|          Total Length         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Identification        |Flags|      Fragment Offset    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Time to Live |Protcol=17(UDP)|          Header Checksum      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       Source IPv4 Address                     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     Destination IPv4 Address                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 UDP Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Source Port = Entropy Value  |  Dest. Port = TBD1/TBD2       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           UDP Length          |        UDP Checksum           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 GRE Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |C| |K|S| Reserved0       | Ver |         Protocol Type         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Checksum (optional)      |       Reserved1 (Optional)    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         Key (optional)                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 Sequence Number (optional)                    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1 UDP+GRE Headers in IPv4

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

 IPv6 Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Version| Traffic Class |           Flow Label                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Payload Length        | NxtHdr=17(UDP)|   Hop Limit   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                                                               +
 |                                                               |
 +                      Outer Source IPv6 Address                +
 |                                                               |
 +                                                               +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                                                               +
 |                                                               |
 +                   Outer Destination IPv6 Address              +
 |                                                               |
 +                                                               +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 UDP Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Source Port = entropy value  |  Dest. Port = TBD1/TBD2       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           UDP Length          |        UDP Checksum           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 GRE Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |C| |K|S| Reserved0       | Ver |         Protocol Type         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Checksum (optional)      |       Reserved1 (Optional)    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        Key (optional)                         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 Sequence Number (optional)                    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2 UDP+GRE Headers in IPv6

The contents of the IP, UDP, and GRE headers that are relevant in this encapsulation are described below.

3.1. IP Header

An encapsulator MUST encode its own IP address as the source IP address and the decapsulator’s IP address as the destination IP address. A sufficiently large value is needed in the IPv4 TTL field or IPv6 Hop Count field to allow delivery of the encapsulated packet to the peer of the encapsulation.

3.2. UDP Header

3.2.1. Source Port

GRE-in-UDP permits the UDP source port value to be used to encode an entropy value. The UDP source port contains a 16-bit entropy value that is generated by the encapsulator to identify a flow for the encapsulated packet. The port value SHOULD be within the ephemeral port range, i.e., 49152 to 65535, where the high order two bits of the port are set to one. This provides fourteen bits of entropy for the inner flow identifier. In the case that an encapsulator is unable to derive flow entropy from the payload header or the entropy usage has to be disabled to meet operational requirements (see Section 7), to avoid reordering with a packet flow, the encapsulator SHOULD use the same UDP source port value for all packets assigned to a flow e.g., the result of an algorithm that perform a hash of the tunnel ingress and egress IP address.

The source port value for a flow set by an encapsulator MAY change over the lifetime of the encapsulated flow. For instance, an encapsulator may change the assignment for Denial of Service (DOS) mitigation or as a means to effect routing through the ECMP network. An encapsulator SHOULD NOT change the source port selected for a flow more than once every thirty seconds.

An IPv6 GRE-in-UDP tunnel endpoint SHOULD copy a flow entropy value in the IPv6 flow label field (requirement 6). This permits network equipment to inspect this value and utilize it during forwarding, e.g. to perform ECMP [RFC6438].

This document places requirements on the generation of the flow entropy value [RFC5405bis] but does not specify the algorithm that an implementation should use to derive this value.

3.2.2. Destination Port

The destination port of the UDP header is set either GRE-in-UDP (TBD1) or GRE-UDP-DTLS (TBD2) (see Section 5).

3.2.3. Checksum

The UDP checksum is set and processed per [RFC768] and [RFC1122] for IPv4, and [RFC2460] for IPv6. Requirements for checksum handling and use of zero UDP checksums are detailed in Section 6.

3.2.4. Length

The usage of this field is in accordance with the current UDP specification in [RFC768]. This length will include the UDP header (eight bytes), GRE header, and the GRE payload (encapsulated packet).

3.3. GRE Header

An encapsulator sets the protocol type (EtherType) of the packet being encapsulated in the GRE Protocol Type field.

An encapsulator MAY set the GRE Key Present, Sequence Number Present, and Checksum Present bits and associated fields in the GRE header as defined by [RFC2784] and [RFC2890]. Usage of the reserved bits, i.e., Reserved0, is specified in [RFC2784].

The GRE checksum MAY be enabled to protect the GRE header and payload. When the UDP checksum is enabled, it protects the GRE payload, resulting in the GRE checksum being mostly redundant. Enabling both checksums may result in unnecessary processing. Since the UDP checksum covers the pseudo-header and the packet payload, including the GRE header and its payload, the UDP checksum SHOULD be used in preference to using the GRE checksum.

An implementation MAY use the GRE keyid to authenticate the encapsulator.(See Security Considerations Section) In this model, a shared value is either configured or negotiated between an encapsulator and decapsulator. When a decapsulator determines a presented keyid is not valid for the source, the packet MUST be dropped.

Although GRE-in-UDP encapsulation protocol uses both UDP header and GRE header, it is one tunnel encapsulation protocol. GRE and UDP headers MUST be applied and removed as a pair at the encapsulation and decapsulation points. This specification does not support UDP encapsulation of a GRE header where that GRE header is applied or

removed at a network node other than the UDP tunnel ingress or egress.

4. Encapsulation Process Procedures

The procedures specified in this section apply to both a default GRE-in-UDP tunnel and a TMCE GRE-in-UDP tunnel.

The GRE-in-UDP encapsulation allows encapsulated packets to be forwarded through "GRE-in-UDP tunnels". The encapsulator MUST set the UDP and GRE header according to Section 3.

Intermediate routers, upon receiving these UDP encapsulated packets, could load balance these packets based on the hash of the five-tuple of UDP packets.

Upon receiving these UDP encapsulated packets, the decapsulator decapsulates them by removing the UDP and GRE headers and then processes them accordingly.

GRE-in-UDP can encapsulate traffic with unicast, IPv4 broadcast, or multicast (see requirement 3 in Section 2.1.1). However a default GRE-in-UDP tunnel MUST NOT be deployed or configured to carry traffic that is not congestion-controlled (See requirement 3 in Section 2.1.1). Entropy may be generated from the header of encapsulated packets at an encapsulator. The mapping mechanism between the encapsulated multicast traffic and the multicast capability in the IP network is transparent and independent of the encapsulation and is otherwise outside the scope of this document.

To provide entropy for ECMP, GRE-in-UDP does not rely on GRE keep-alive. It is RECOMMENED not to use GRE keep-alive in the GRE-in-UDP tunnel. This aligns with middlebox traversal guidelines in Section 3.5 of [RFC5405bis].

4.1. MTU and Fragmentation

Regarding packet fragmentation, an encapsulator/decapsulator SHOULD perform fragmentation before the encapsulation. The size of fragments SHOULD be less or equal to the Path MTU (PMTU) associated with the path between the GRE ingress and the GRE egress tunnel endpoints minus the GRE and UDP overhead, assuming the egress MTU for reassembled packets is larger than PMTU. When applying payload fragmentation, the UDP checksum MUST be used so that the receiving endpoint can validate reassembly of the fragments; the same source UDP port SHOULD be used for all packet fragments to ensure the transit routers will forward the fragments on the same path.

If the operator of the transit network supporting the tunnel is able to control the payload MTU size, the MTU SHOULD be configured to avoid fragmentation, i.e., sufficient for the largest supported size of packet, including all additional bytes introduced by the tunnel overhead [RFC5405bis].

4.2. Differentiated Services and ECN Marking

To ensure that tunneled traffic receives the same treatment over the IP network as traffic that is not tunneled, prior to the encapsulation process, an encapsulator processes the tunneled IP packet headers to retrieve appropriate parameters for the encapsulating IP packet header such as DiffServ [RFC2983]. Encapsulation end points that support Explicit Congestion Notification (ECN) must use the method described in [RFC6040] for ECN marking propagation. The congestion control process is outside of the scope of this document.

Additional information on IP header processing is provided in Section 3.1.

5. Use of DTLS

Datagram Transport Layer Security (DTLS) [RFC6347] can be used for application security and can preserve network and transport layer protocol information. Specifically, if DTLS is used to secure the GRE-in-UDP tunnel, the destination port of the UDP header MUST be set to an IANA-assigned value (TBD2) indicating GRE-in-UDP with DTLS, and that UDP port MUST NOT be used for other traffic. The UDP source port field can still be used to add entropy, e.g., for load-sharing purposes. DTLS applies to a default GRE-in-UDP tunnel and a TMCE GRE-in-UDP tunnel.

Use of DTLS is limited to a single DTLS session for any specific tunnel encapsulator/decapsulator pair (identified by source and destination IP addresses). Both IP addresses MUST be unicast addresses - multicast traffic is not supported when DTLS is used. A GRE-in-UDP tunnel decapsulator that supports DTLS is expected to be able to establish DTLS sessions with multiple tunnel encapsulators, and likewise a GRE-in-UDP tunnel encapsulator is expected to be able to establish DTLS sessions with multiple decapsulators. Different source and/or destination IP addresses will be involved (see Section 6.2) for discussion of one situation where use of different source IP addresses is important.

When DTLS is used for a GRE-in-UDP tunnel, if a packet is received from the tunnel and that packet is not protected by the DTLS session

or part of DTLS negotiation (e.g., a DTLS handshake message [RFC6347]), the tunnel receiver MUST discard that packet and SHOULD log that discard event and information about the discarded packet.

DTLS SHOULD be used for a GRE-in-UDP tunnel to meet security requirements of the original traffic that is delivered by a GRE-in-UDP tunnel. There are cases where no additional security is required, e.g., the traffic to be encapsulated is already encrypted or the tunnel is deployed within an operationally secured network. Use of DTLS for a GRE-in-UDP tunnel requires both tunnel endpoints to configure use of DTLS.

6. UDP Checksum Handling

6.1. UDP Checksum with IPv4

For UDP in IPv4, when a non-zero UDP checksum is used, the UDP checksum MUST be processed as specified in [RFC768] and [RFC1122] for both transmit and receive. The IPv4 header includes a checksum that protects against mis-delivery of the packet due to corruption of IP addresses. The UDP checksum potentially provides protection against corruption of the UDP header, GRE header, and GRE payload. Disabling the use of checksums is a deployment consideration that should take into account the risk and effects of packet corruption.

When a decapsulator receives a packet, the UDP checksum field MUST be processed. If the UDP checksum is non-zero, the decapsulator MUST verify the checksum before accepting the packet. By default a decapsulator SHOULD accept UDP packets with a zero checksum. A node MAY be configured to disallow zero checksums per [RFC1122]; this may be done selectively, for instance disallowing zero checksums from certain hosts that are known to be sending over paths subject to packet corruption. If verification of a non-zero checksum fails, a decapsulator lacks the capability to verify a non-zero checksum, or a packet with a zero-checksum was received and the decapsulator is configured to disallow, the packet MUST be dropped and an event MAY be logged.

6.2. UDP Checksum with IPv6

For UDP in IPv6, the UDP checksum MUST be processed as specified in [RFC768] and [RFC2460] for both transmit and receive.

When UDP is used over IPv6, the UDP checksum is relied upon to protect both the IPv6 and UDP headers from corruption. As such, A default GRE-in-UDP Tunnel MUST perform UDP checksum; A TMCE GRE-in-UDP Tunnel MAY be configured with the UDP zero-checksum mode if the

traffic-managed controlled environment or a set of closely cooperating traffic-managed controlled environments (such as by network operators who have agreed to work together in order to jointly provide specific services) meet at least one of following conditions:

a. It is known (perhaps through knowledge of equipment types and lower layer checks) that packet corruption is exceptionally unlikely and where the operator is willing to take the risk of undetected packet corruption.

b. It is judged through observational measurements (perhaps of historic or current traffic flows that use a non-zero checksum) that the level of packet corruption is tolerably low and where the operator is willing to take the risk of undetected packet corruption.

c. Carrying applications that are tolerant of mis-delivered or corrupted packets (perhaps through higher layer checksum, validation, and retransmission or transmission redundancy) where the operator is willing to rely on the applications using the tunnel to survive any corrupt packets.

The following requirements apply to a TMCE GRE-in-UDP tunnel that uses UDP zero-checksum mode:

a. Use of the UDP checksum with IPv6 MUST be the default configuration of all GRE-in-UDP tunnels.

b. The GRE-in-UDP tunnel implementation MUST comply with all requirements specified in Section 4 of [RFC6936] and with requirement 1 specified in Section 5 of [RFC6936].

c. The tunnel decapsulator SHOULD only allow the use of UDP zero-checksum mode for IPv6 on a single received UDP Destination Port regardless of the encapsulator. The motivation for this requirement is possible corruption of the UDP Destination Port, which may cause packet delivery to the wrong UDP port. If that other UDP port requires the UDP checksum, the mis-delivered packet will be discarded.

d. It is RECOMMENDED that the UDP zero-checksum mode for IPv6 is only enabled for certain selected source addresses. The tunnel decapsulator MUST check that the source and destination IPv6 addresses are valid for the GRE-in-UDP tunnel on which the packet was received if that tunnel uses UDP zero-checksum mode and discard any packet for which this check fails.

e. The tunnel encapsulator SHOULD use different IPv6 addresses for each GRE-in-UDP tunnel that uses UDP zero-checksum mode regardless of the decapsulator in order to strengthen the decapsulator’s check of the IPv6 source address (i.e., the same IPv6 source address SHOULD NOT be used with more than one IPv6 destination address, independent of whether that destination address is a unicast or multicast address). When this is not possible, it is RECOMMENDED to use each source IPv6 address for as few UDP zero-checksum mode GRE-in-UDP tunnels as is feasible.

f. When any middlebox exists on the path of a GRE-in-UDP tunnel, it is RECOMMENDED to use the default mode, i.e. use UDP checksum, to reduce the chance that the encapsulated packets will be dropped.

g. Any middlebox that allows the UDP zero-checksum mode for IPv6 MUST comply with requirement 1 and 8-10 in Section 5 of [RFC6936].

h. Measures SHOULD be taken to prevent IPv6 traffic with zero UDP checksums from "escaping" to the general Internet; see Section 8 for examples of such measures.

i. IPv6 traffic with zero UDP checksums MUST be actively monitored for errors by the network operator. For example, the operator may monitor Ethernet layer packet error rates.

j. If a packet with a non-zero checksum is received, the checksum MUST be verified before accepting the packet. This is regardless of whether the tunnel encapsulator and decapsulator have been configured with UDP zero-checksum mode.

The above requirements do not change either the requirements specified in [RFC2460] as modified by [RFC6935] or the requirements specified in [RFC6936].

The requirement to check the source IPv6 address in addition to the destination IPv6 address, plus the strong recommendation against reuse of source IPv6 addresses among GRE-in-UDP tunnels collectively provide some mitigation for the absence of UDP checksum coverage of the IPv6 header. A traffic-managed controlled environment that satisfies at least one of three conditions listed at the beginning of this section provides additional assurance.

A GRE-in-UDP tunnel is suitable for transmission over lower layers in the traffic-managed controlled environments that are allowed by the exceptions stated above and the rate of corruption of the inner

IP packet on such networks is not expected to increase by comparison to GRE traffic that is not encapsulated in UDP. For these reasons, GRE-in-UDP does not provide an additional integrity check except when GRE checksum is used when UDP zero-checksum mode is used with IPv6, and this design is in accordance with requirements 2, 3 and 5 specified in Section 5 of [RFC6936].

Generic Router Encapsulation (GRE) does not accumulate incorrect transport layer state as a consequence of GRE header corruption. A corrupt GRE packet may result in either packet discard or forwarding of the packet without accumulation of GRE state. Active monitoring of GRE-in-UDP traffic for errors is REQUIRED as occurrence of errors will result in some accumulation of error information outside the protocol for operational and management purposes. This design is in accordance with requirement 4 specified in Section 5 of [RFC6936].

The remaining requirements specified in Section 5 of [RFC6936] are not applicable to GRE-in-UDP. Requirements 6 and 7 do not apply because GRE does not include a control feedback mechanism. Requirements 8-10 are middlebox requirements that do not apply to GRE-in-UDP tunnel endpoints (see Section 7.1 for further middlebox discussion).

It is worth mentioning that the use of a zero UDP checksum should present the equivalent risk of undetected packet corruption when sending similar packet using GRE-in-IPv6 without UDP [RFC7676] and without GRE checksums.

In summary, a TMCE GRE-in-UDP Tunnel is allowed to use UDP-zero-checksum mode for IPv6 when the conditions and requirements stated above are met. Otherwise the UDP checksum need to be used for IPv6 as specified in [RFC768] and [RFC2460]. Use of GRE checksum is RECOMMENED when the UDP checksum is not used.

7. Middlebox Considerations

Many middleboxes read or update UDP port information of the packets that they forward. Network Address/Port Translator (NAPT) is the most commonly deployed Network Address Translation (NAT) device [RFC4787]. An NAPT device establishes a NAT session to translate the {private IP address, private source port number} tuple to a {public IP address, public source port number} tuple, and vice versa, for the duration of the UDP session. This provides a UDP application with the "NAT-pass-through" function. NAPT allows multiple internal hosts to share a single public IP address. The port number, i.e., the UDP Source Port number, is used as the demultiplexer of the multiple internal hosts. However, the above NAPT behaviors conflict

with the behavior a GRE-in-UDP tunnel that is configured to use the UDP source port value to provide entropy.

A GRE-in-UDP tunnel is unidirectional; the tunnel traffic is not expected to be returned back to the UDP source port values used to generate entropy. However some middleboxes (e.g., firewall) assume that bidirectional traffic uses a common pair of UDP ports. This assumption also conflicts with the use of the UDP source port field as entropy.

Hence, use of the UDP source port for entropy may impact middleboxes behavior. If a GRE-in-UDP tunnel is expected to be used on a path with a middlebox, the tunnel can be configured to either disable use of the UDP source port for entropy or to configure middleboxes to pass packets with UDP source port entropy.

7.1. Middlebox Considerations for Zero Checksums

IPv6 datagrams with a zero UDP checksum will not be passed by any middlebox that updates the UDP checksum field or simply validates the checksum based on [RFC2460], such as firewalls. Changing this behavior would require such middleboxes to be updated to correctly handle datagrams with zero UDP checksums. The GRE-in-UDP encapsulation does not provide a mechanism to safely fall back to using a checksum when a path change occurs redirecting a tunnel over a path that includes a middlebox that discards IPv6 datagrams with a zero UDP checksum. In this case the GRE-in-UDP tunnel will be black-holed by that middlebox.

As such, when any middlebox exists on the path of GRE-in-UDP tunnel, use of the UDP checksum is RECOMMENDED to increase the probability of successful transmission of GRE-in-UDP packets. Recommended changes to allow firewalls and other middleboxes to support use of an IPv6 zero UDP checksum are described in Section 5 of [RFC6936].

8. Congestion Considerations

Section 3.1.9 of [RFC5405bis] discusses the congestion considerations for design and use of UDP tunnels; this is important because other flows could share the path with one or more UDP tunnels, necessitating congestion control [RFC2914] to avoid distractive interference.

Congestion has potential impacts both on the rest of the network containing a UDP tunnel, and on the traffic flows using the UDP tunnels. These impacts depend upon what sort of traffic is carried over the tunnel, as well as the path of the tunnel. The GRE-in-UDP

tunnel protocol does not provide any congestion control and GRE-in-UDP packets are regular UDP packets. Therefore, a GRE-in-UDP tunnel MUST NOT be deployed to carry non-congestion controlled traffic over the Internet [RFC5405bis].

Within a TMCE network, GRE-in-UDP tunnels are appropriate for carrying traffic that is not known to be congestion controlled. For example, a GRE-in-UDP tunnel may be used to carry Multiprotocol Label Switching (MPLS) traffic such as pseudowires or VPNs where specific bandwidth guarantees are provided to each pseudowire or VPN. In such cases, operators of TMCE networks avoid congestion by careful provisioning of their networks, rate limiting of user data traffic, and traffic engineering according to path capacity.

When a GRE-in-UDP tunnel carries traffic that is not known to be congestion controlled in a TMCE network, the tunnel MUST be deployed entirely within that network, and measures SHOULD be taken to prevent the GRE-in-UDP traffic from "escaping" the network to the general Internet, e.g.:

- o Physical or logical isolation of the links carrying GRE-in-UDP from the general Internet.

- o Deployment of packet filters that block the UDP ports assigned for GRE-in-UDP.

- o Imposition of restrictions on GRE-in-UDP traffic by software tools used to set up GRE-in-UDP tunnels between specific end systems (as might be used within a single data center) or by tunnel ingress nodes for tunnels that don’t terminate at end systems.

9. Backward Compatibility

In general, tunnel ingress routers have to be upgraded in order to support the encapsulations described in this document.

No change is required at transit routers to support forwarding of the encapsulation described in this document.

If a tunnel endpoint (a host or router) that is intended for use as a decapsulator does not support or enable the GRE-in-UDP encapsulation described in this document, that endpoint will not listen on the destination port assigned to the GRE-encapsulation (TBD1 and TBD2). In these cases, the endpoint will perform normal UDP processing and respond to an encapsulator with an ICMP message

indicating "port unreachable" according to [RFC792]. Upon receiving this ICMP message, the node MUST NOT continue to use GRE-in-UDP encapsulation toward this peer without management intervention.

10. IANA Considerations

IANA is requested to make the following allocations:

One UDP destination port number for the indication of GRE,

```
Service Name: GRE-in-UDP
Transport Protocol(s): UDP
Assignee: IESG <iesg@ietf.org>
Contact: IETF Chair <chair@ietf.org>
Description: GRE-in-UDP Encapsulation
Reference: [This.I-D]
Port Number: TBD1
Service Code: N/A
Known Unauthorized Uses: N/A
Assignment Notes: N/A
```

Editor Note: replace "TBD1" in section 3 and 9 with IANA assigned number.

One UDP destination port number for the indication of GRE with DTLS,

```
Service Name: GRE-UDP-DTLS
Transport Protocol(s): UDP
Assignee: IESG <iesg@ietf.org>
Contact: IETF Chair <chair@ietf.org>
Description: GRE-in-UDP Encapsulation with DTLS
Reference: [This.I-D]
Port Number: TBD2
Service Code: N/A
Known Unauthorized Uses: N/A
Assignment Notes: N/A
```

Editor Note: replace "TBD2" in section 3, 5, and 9 with IANA assigned number.

11. Security Considerations

GRE-in-UDP encapsulation does not affect security for the payload protocol. The security considerations for GRE apply to GRE-in-UDP, see [RFC2784].

To secure traffic carried by a GRE-in-UDP tunnel, DTLS SHOULD be used as specified in Section 5.

In the case that UDP source port for entropy usage is disabled, a random port SHOULD be selected in order to minimize the vulnerability to off-path attacks [RFC6056]. The random port may also be periodically changed to mitigate certain denial of service attacks as mentioned in Section 3.2.1.

Using one standardized value as the UDP destination port to indicate an encapsulation may increase the vulnerability of off-path attack. To overcome this, an alternate port may be agreed upon to use between an encapsulator and decapsulator [RFC6056]. How the encapsulator end points communicate the value is outside scope of this document.

This document does not require that a decapsulator validates the IP source address of the tunneled packets (with the exception that the IPv6 source address MUST be validated when UDP zero-checksum mode is used with IPv6), but it should be understood that failure to do so presupposes that there is effective destination-based (or a combination of source-based and destination-based) filtering at the boundaries.

Corruption of GRE headers can cause security concerns for applications that rely on the GRE key field for traffic separation or segregation. When the GRE key field is used for this purpose such as an application of a Network Virtualization Using Generic Routing Encapsulation (NVGRE) [RFC7637], GRE header corruption is a concern. In such situations, at least one of the UDP and GRE checksums MUST be used for both IPv4 and IPv6 GRE-in-UDP tunnels.

12. Acknowledgements

Authors like to thank Vivek Kumar, Ron Bonica, Joe Touch, Ruediger Geib, Lars Eggert, Lloyd Wood, Bob Briscoe, Rick Casarez, Jouni Korhonen, Kathleen Moriarty, Ben Campbell, and many others for their review and valuable input on this draft.

Thank Donald Eastlake, Eliot Lear, Martin Stiemerling, and Spencer Dawkins for their detail reviews and valuable suggestions in WGLC and IESG process.

Thank the design team led by David Black (members: Ross Callon, Gorry Fairhurst, Xiaohu Xu, Lucy Yong) to efficiently work out the descriptions for the congestion considerations and IPv6 UDP zero checksum.

Thank David Black and Gorry Fairhurst for their great help in document content and editing.

13. Contributors

The following people all contributed significantly to this document and are listed below in alphabetical order:

David Black
EMC Corporation
176 South Street
Hopkinton, MA 01748
USA

Email: david.black@emc.com

Ross Callon
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: rcallon@juniper.net

John E. Drake
Juniper Networks

Email: jdrake@juniper.net

Gorry Fairhurst
University of Aberdeen

Email: gorry@erg.abdn.ac.uk

Yongbing Fan
China Telecom
Guangzhou, China.
Phone: +86 20 38639121

Email: fanyb@gsta.com

Adrian Farrel
Juniper Networks

Email: adrian@olddog.co.uk

Vishwas Manral
Hewlett-Packard Corp.
3000 Hanover St, Palo Alto.

Email: vishwas.manral@hp.com

Carlos Pignataro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, NC 27709 USA

Email: cpignata@cisco.com

14. References

14.1. Normative References

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC1122] Braden, R., "Requirements for Internet Hosts -- Communication Layers", RFC1122, October 1989.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC2119, March 1997.

[RFC2474] Nichols K., Blake S., Baker F., Black D., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", December 1998.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC2890, September 2000.

[RFC5405bis] Eggert, L., "Unicast UDP Usage Guideline for Application Designers", draft-ietf-tsvwg-rfc5405bis, work in progress.

[RFC6040] Briscoe, B., "Tunneling of Explicit Congestion Notification", RFC6040, November 2010.

[RFC6347] Rescoria, E., Modadugu, N., "Datagram Transport Layer Security Version 1.2", RFC6347, 2012.

[RFC6438] Carpenter, B., Amante, S., "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in tunnels", RFC6438, November, 2011.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, April 2013.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.

14.2. Informative References

[RFC792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC793] DARPA, "Transmission Control Protocol", RFC793, September 1981.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2914] Floyd, S., "Congestion Control Principles", RFC2914, September 2000.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC2983, October 2000.

[RFC4787] Audet, F., et al, "network Address Translation (NAT) Behavioral Requirements for Unicast UDP", RFC4787, January 2007.

[RFC6056] Larsen, M. and Gont, F., "Recommendations for Transport-Protocol Port Randomization", RFC6056, January 2011.

[RFC6438] Carpenter, B., Amante, S., "Using the Ipv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC6438, November 2011.

[RFC7042] Eastlake 3rd, D. and Abley, J., "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameter", RFC7042, October 2013.

[RFC7637] Garg, P. and Wang, Y., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC7637, September 2015.

[RFC7676] Pignataro, C., Bonica, R., Krishnan, S., "IPv6 Support for Generic Routing Encapsulation (GRE)", RFC7676, October 2015.

15. Authors' Addresses

Lucy Yong
Huawei Technologies, USA

Email: lucy.yong@huawei.com

Edward Crabbe
Oracle

Email: edward.crabbe@gmail.com

Xiaohu Xu
Huawei Technologies,
Beijing, China

Email: xuxiaohu@huawei.com

Tom Herbert
Facebook
1 Hacker Way
Menlo Park, CA
Email : tom@herbertland.com

Network WG
Internet-Draft
Expires: August 25, 2013
Intended Status: Standards Track (PS)
Updates: RFC 2205, 2210, & 4495 (if published as an RFC)

James Polk
Subha Dhesikan
Cisco Systems

February 25, 2013

Integrated Services (IntServ) Extension to Allow Signaling of Multiple Traffic Specifications and Multiple Flow Specifications in RSVPv1

draft-ietf-tsvwg-intserv-multiple-tspec-02

Abstract

This document defines extensions to Integrated Services (IntServ) allowing multiple traffic specifications and multiple flow specifications to be conveyed in the same Resource Reservation Protocol (RSVPv1) reservation message exchange. This ability helps optimize an agreeable bandwidth through a network between endpoints in a single round trip.

Status of this Memo

Copyright Notice

Table of Contents

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC 2119].

1. Introduction

This document defines how Integrated Services (IntServ) [RFC2210] includes multiple traffic specifications and multiple flow specifications in the same Resource Reservation Protocol (RSVPv1) [RFC2205] message. This ability helps optimize an agreeable bandwidth through a network between endpoints in a single round trip.

There is a separation of function between RSVP and IntServ, in which RSVP does not define the internal objects to establish controlled load or guarantee services. These are generally left to be opaque in RSVP. At the same time, IntServ does not require that RSVP be the only reservation protocol for transporting both the controlled load or guaranteed service objects - but RSVP does often carry the objects anyway. This makes the two independent - yet related in usage, but are also frequently talked about as if they are one and the same. They are not.

The 'traffic specification' contains the traffic characteristics of a sender's data flow and is a required object in a PATH message. The TSPEC object is defined in RFC 2210 to convey the traffic specification from the sender and is opaque to RSVP. The ADSPEC object - for 'advertising specification' - is used to gather information along the downstream data path to aid the receiver in the computation of QoS properties of this data path. The ADSPEC is also opaque to RSVP and is defined in RFC 2210. Both of these IntServ objects are part of the Sender Descriptor [RFC2205].

Once the Sender Descriptor is received at its destination node, after having traveled through the network of routers, the SENDER_TSPEC information is matched with the information gathered in the ADSPEC, if present, about the data path. Together, these two objects help the receiver build its flow specification (encoded in the FLOWSPEC object) for the RESV message. The RESV message establishes the reservation through the network of routers on the data path established by the PATH message. If the ADSPEC is not present in the Sender_Descriptor, it cannot aid the receiver in building the flow specification.

The SENDER_TSPEC is not changed in transit between endpoints (i.e., there are no bandwidth request adjustments along the way). However, the ADSPEC is changed, based on the conditions experienced through the network (i.e., bandwidth availability within each router) as the RSVP message travels hop-by-hop.

Today, real-time applications have evolved such that they are able to dynamically adapt to available bandwidth, not only by dropping and adding layers, but also by reducing frame rates and resolution. It is therefore limiting to have a single bandwidth request in Integrated Services, and by extension, RSVP.

With only one traffic specification in a PATH message and only one flow specification in a RESV message (with some styles of reservations a RESV message may actually contain multiple flow specifications, but then there is only one per sender), applications will either have to give up altogether on session establishment in case of failure of the reservation establishment for the highest "bandwidth or will have to resort to multiple successive RSVP signaling attempts in a trial-and-error manner until they finally establish the reservation a lower "bandwidth". These multiple signaling round-trip would affect the session establishment time and in turn would negatively impact the end user experience.

The objective of this document is to avoid such roundtrips as well as allow applications to successfully receive some level of bandwidth allotment that it can use for its sessions.

While the ADSPEC provides an indication of the bandwidth available along the path and can be used by the receiver in creating the FLOWSPEC, it does not prevent failures or multiple round-trips as described above. The intermediary routers provide a best attempt estimate of available bandwidth in the ADSPEC object. However, it does not take into account external policy considerations (RFC 2215). In addition, the available bandwidth at the time of creating the ADSPEC may not be available at the time of an actual request in an RESV message. These reasons may cause the RESV message to be rejected. Therefore, the ADSPEC object cannot, by itself, satisfy the requirements of the current generations of real-time applications.

It needs to be noted that the ADSPEC is unchanged by this new mechanism. If ADSPEC is included in the PATH message, it is suggested that the receiver use this object in determining the flow specification.

This document creates a means for conveying more than one "bandwidth" within the same RSVP reservation set-up (both PATH and RESV) messages to optimize the determination of an agreed upon bandwidth for this reservation. Allowing multiple traffic specifications within the same PATH message allows the sender to communicate to the receiver multiple "bandwidths" that match the different sending rates that the sender is capable of transmitting at. This allows the receiver to convey this multiple "bandwidths" in the RESV so those can be considered when RSVP makes the actual reservation admission into the network. This allows the applications to dynamically adapt their data stream to available network resources.

The concept of RSVP signaling is shown in a single direction below, in Figure 1. Although the TSPEC is opaque to RSVP, it is shown along with the RSVP messages for completeness. The RSVP messages themselves need not be the focus of the reader. Instead, the

number of round trips it takes to establish a reservation is the focus here.

```
   Sender         Rtr-1      Rtr-2 ...  Rtr-N         Receiver
     |              |          |          |               |
     |           PATH (with a TSPEC & ADSPEC)             |
     |------------->|--------->|----//--->|-------------->|
     |              |          |          |               |
     |                            RESV (with a FLOWSPEC)  |
     |<-------------|<---------|<---//----|<--------------|
     |              |          |          |               |
```

Figure 1. Concept of RSVP in a Single Direction

Figure 1 shows a successful one-way reservation using RSVP and IntServ.

Figure 2 shows a scenario where the RESV message, containing a FLOWSPEC, which is generated by the Receiver, after considering both the Sender TSPEC and the ADSPEC, is rejected by an intermediary router.

```
   Sender         Rtr-1      Rtr-2 ...  Rtr-N         Receiver
     |              |          |          |               |
     |           PATH (with 1 TSPEC wanting 12Mbps)       |
     |------------->|--------->|----//--->|-------------->|
     |              |          |          |               |
     |              RESV (with 1 FLOWSPEC wanting 12Mbps) |
     |              |          X <--//----|<--------------|
     |              |          |          |               |
     |           ResvErr (with Admission control Error=2) |
     |              |          |----//--->|-------------->|
     |              |          |          |               |
```

Figure 2. Concept of RSVP Rejection due to Limited Bandwidth

The scenario above is where multiple TSPEC and multiple FLOWSPEC optimization helps. The Sender may support multiple bandwidths for a given application (i.e., more than one codec for voice or video) and therefore might want to establish a reservation with the highest (or best) bandwidth that the network can provide for a particular codec.

For example, bandwidths of:

```
    12Mbps,
     4Mbps, and
     1.5Mbps
```

for the three video codecs the Sender supports.

This document will discuss the overview of the proposal to include multiple TSPECs and FLOWSPECs RSVP in section 2. In section 3, the overview of the entire solution is provided. This section also contains the new parameters which are defined in this document. The multiple TSPECs in a PATH message and the multiple FLOWSPEC in a RESV message, both for controlled load and guaranteed service are described in this section. Section 4 will cover the rules of usage of this IntServ extension. This section contains how this document needs to extend the scenario of when a router in the middle of a reservation cannot accept a preferred bandwidth (i.e., FLOWSPEC), meaning previous routers that accepted that greater bandwidth now have too much bandwidth reserved. This requires an extension to RFC 4495 (RSVP Bandwidth Reduction) to cover reservations being established, as well as existing reservations. Section 4 also includes the merging rules.

2. Overview of Proposal for Including Multiple TSPECs and FLOWSPECS

Presently, this is the format of a PATH message [RFC2205]:

```
<PATH Message> ::= <Common Header> [ <INTEGRITY> ]

                   <SESSION> <RSVP_HOP>

                   <TIME_VALUES>

                  [ <POLICY_DATA> ... ]

                  [ <sender descriptor> ]

<sender descriptor> ::= <SENDER_TEMPLATE> <SENDER_TSPEC>
                                           ^^^^^^^^^^^^
                  [ <ADSPEC> ]
```

where the SENDER_TSPEC object contains a single traffic specification.

For the PATH message, the focus of this document is to modify the <sender_descriptor> in such a way to include more than one traffic specification. This solution does this by retaining the existing SENDER_TSPEC object above, highlighted by the '^^^^' characters, and complementing it with a new optional MULTI_TSPEC object to convey additional traffic specifications in this PATH message. No other object within the PATH message is affected by this IntServ extension.

This extension modifies the sender descriptor by specifically augmenting it to allow an optional <MULTI_TSPEC> object after the optional <ADSPEC>, as shown below.

```
<sender descriptor> ::= <SENDER_TEMPLATE> <SENDER_TSPEC>

                        [ <ADSPEC> ]  [ <MULTI_TSPEC> ]
                                        ^^^^^^^^^^^^
```

As can be seen above, the MULTI_TSPEC is in addition to the SENDER_TSPEC - and is only to be used, per this extension, when more than one TSPEC is to be included in the PATH message.

Here is another way of looking at the proposal choices:

```
+---------------------+
|   Existing TSPEC    |
|                     |
|    +----------+     |
|    |  TSPEC1  |     |
|    +----------+     |
|                     |
+---------------------+

+---------------------+
|  Additional TSPECs  |
|                     |
|  +---------------+  |
|  |  MULTI_TSPEC  |  |
|  |    Object     |  |
|  |  +--------+   |  |
|  |  | TSPEC2 |   |  |
|  |  +--------+   |  |
|  |  +--------+   |  |
|  |  | TSPEC3 |   |  |
|  |  +--------+   |  |
|  |  +--------+   |  |
|  |  | TSPEC4 |   |  |
|  |  +--------+   |  |
|  +---------------+  |
|                     |
+---------------------+
```

Figure 3. Encoding of Multiple Traffic Specifications in the TSPEC and MULTI_TSPEC objects

This solution is backwards compatible with existing implementations of [RFC2205] and [RFC2210], as the multiple TSPECs and FLOWSPECs are inserted as optional objects and such objects do not need to be processed, especially if they are not understood.

This solution defines a similar approach for encoding multiple flow specifications in the RESV message. Flow specifications beyond the first one can be encoded in a new "MULTI_FLOWSPEC" object contained

in the RESV message.

In this proposal, the original SENDER_TSPEC and the FLOWSPEC are left untouched, allowing routers not supporting this extension to process the PATH and the RESV message without issue. Two new additional objects are defined in this document. They are the MULTI_TSPEC and the MULTI_FLOWSPEC for the PATH and the RESV message, respectively. The additional TSPECs (in the new MULTI_TSPEC Object) are included in the PATH and the additional FLOWSPECS (in the new MULTI_FLOWSPEC Object) are included in the RESV message as new (optional) objects. These additional objects will have a class number of 11bbbbbb, allowing older routers to ignore the object(s) and forward each unexamined and unchanged, as defined in section 3.10 of [RFC 2205].

NOTE: it is important to emphasize here that including more than one FLOWSPEC in the RESV message does not cause more than one FLOWSPEC to be granted. This document requires that the receiver arrange these multiple FLOWSPECs in the order of preference according to the order remaining from the MULTI_TSPECs in the PATH message. The benefit of this arrangement is that RSVP does not have to process the rest of the FLOWSPEC if it can admit the first one.

3. MULTI_TSPEC and MULTI_FLOWSPEC Solution

For the Sender Descriptor within the PATH message, the original TSPEC remains where it is, and is untouched by this IntServ extension. What is new is the use of a new <MULTI_TSPEC> object inside the sender descriptor as shown here:

```
<sender descriptor> ::= <SENDER_TEMPLATE> <SENDER_TSPEC>

                        [ <ADSPEC> ]  [ <MULTI_TSPEC> ]
                                        ^^^^^^^^^^^^
```

The preferred order of TSPECs sent by the sender is this:

- preferred TSPEC is in the original SENDER_TSPEC

- the next in line preferred TSPEC is the first TSPEC in the MULTI_TSPEC object

- the next in line preferred TSPEC is the second TSPEC in the MULTI_TSPEC object

- and so on...

The composition of the flow descriptor list in a Resv message depends upon the reservation style. Therefore, the following shows

the inclusion of the MULTI_FLOWSPEC object with each of the styles:

```
WF Style:
          <flow descriptor list> ::=  <WF flow descriptor>

          <WF flow descriptor> ::= <FLOWSPEC> [MULTI_FLOWSPEC]

FF style:
          <flow descriptor list> ::=

                    <FLOWSPEC>  <FILTER_SPEC>  [MULTI_FLOWSPEC] |

                    <flow descriptor list> <FF flow descriptor>

          <FF flow descriptor> ::=

                    [ <FLOWSPEC> ] <FILTER_SPEC> [MULTI_FLOWSPEC]

SE style:
          <flow descriptor list> ::= <SE flow descriptor>

          <SE flow descriptor> ::=

                   <FLOWSPEC> <filter spec list> [MULTI_FLOWSPEC]

          <filter spec list> ::=  <FILTER_SPEC>

                              |  <filter spec list> <FILTER_SPEC>
```

3.1 New MULTI_TSPEC and MULTI-RSPEC Parameters

This extension to Integrated Services defines two new parameters They are:

1. <parameter name> Multiple_Token_Bucket_Tspec, with a parameter number of 125.

2. <parameter name> Multiple_Guaranteed_Service_RSpec with a parameter number of 124

These are IANA registered in this document.

The original SENDER_TSPEC and FLOWSPEC for Controlled Service maintain the <parameter name> of Token_Bucket_Tspec with a parameter number of 127. The original FLOWSPEC for Guaranteed Service maintains the <parameter name> of Guaranteed_Service_RSpec with a parameter number of 130.

3.2 Multiple TSPEC in a PATH Message

Here is the object from [RFC2210]. It is used as a SENDER_TSPEC in a PATH message:

```
     31           24 23           16 15            8 7             0
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 1  | 0 (a) |    reserved           |             7 (b)             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 2  |    X  (c)     |0| reserved    |             6 (d)             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 3  |   127 (e)     |    0 (f)      |             5 (g)             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 4  |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 5  |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 6  |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 7  |  Minimum Policed Unit [m] (32-bit integer)                    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 8  |  Maximum Packet Size [M]  (32-bit integer)                    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4. SENDER_TSPEC in PATH

(a) - Message format version number (0)
(b) - Overall length (7 words not including header)
(c) - Service header, service number
  - '1' (Generic information) if in a PATH message;
(d) - Length of service data, 6 words not including per-service header
(e) - Parameter ID, parameter 127 (Token Bucket TSpec)
(f) - Parameter 127 flags (none set)
(g) - Parameter 127 length, 5 words not including per-service header

For completeness, Figure 4 is included in its original form for backwards compatibility reasons, as if there were only 1 TSPEC in the PATH. What is new when there are more than one TSPEC in this reservation message is the new MULTI_TSPEC object in Figure 5 containing, for example, 3 (Multiple_Token_Bucket_Tspec) TSPECs in a PATH message.

```
     31           24 23           16 15            8 7             0
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 1  | 0 (a) |    reserved           |            19 (b)             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 2  |    5  (c)     |0| reserved    |            18 (d)             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 3  |   125 (e)     |    0 (f)      |             5 (g)             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 4  |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
  5   |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  6   |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  7   |  Minimum Policed Unit [m] (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  8   |  Maximum Packet Size [M]  (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  9   |   125 (e)     |    0 (f)      |             5 (g)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 10   |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 11   |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 12   |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 13   |  Minimum Policed Unit [m] (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 14   |  Maximum Packet Size [M]  (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 15   |   125 (e)     |    0 (f)      |             5 (g)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 16   |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 17   |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 18   |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 19   |  Minimum Policed Unit [m] (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 20   |  Maximum Packet Size [M]  (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5. MULTI_TSPEC Object

(a) - Message format version number (0)
(b) - Overall length (19 words not including header)
(c) - Service header, service number 5 (Controlled-Load)
(d) - Length of service data, 18 words not including per-service header
(e) - Parameter ID, parameter 125 (Multiple Token Bucket TSpec)
(f) - Parameter 125 flags (none set)
(g) - Parameter 125 length, 5 words not including per-service header

Figure 5 shows the 2nd through Nth TSPEC in the PATH in the preferred order. The message format (a) remains the same for a second TSPEC and for other additional TSPECs.

The Overall Length (b) includes all the TSPECs within this object, plus the 2nd Word (containing fields (c) and (d)), which MUST NOT be repeated. The service header fields (e),(f) and(g) are repeated for

each TSPEC.

The Service header, here service number 5 (Controlled-Load) MUST remain the same.

Each TSPEC is six 32-bit Words long (the per-service header plus the 5 values that are 1 Word each in length), therefore the length is in 6 Word increments for each additional TSPEC. Case in point, from the above Figure 5, Words 3-8 are the first TSPEC (2nd preferred), Words 9-14 are the next TSPEC (3rd preferred), and Words 15-20 are the final TSPEC (and 4th preferred) in this example of 3 TSPECs in this MULTI_TSPEC object. There is no limit placed on the number of TSPECs a MULTI_TSPEC object can have. However, it is RECOMMENDED to administratively limit the number of TSPECs in the MULTI_TSPEC object to 9 (making for a total of 10 in the PATH message).

The TSPECS are included in the order of preference by the message generator (PATH) and MUST be maintained in that order all the way to the Receiver. The order of TSPECs that are still grantable, in conjunction with the ADSPEC at the Receiver, MUST retain that order in the FLOWSPEC and MULTI_FLOWSPEC objects.

3.3 Multiple FLOWSPEC for Controlled-Load service

The format of an RSVP FLOWSPEC object requesting Controlled-Load service is the same as the one used for the SENDER_TSPEC given in Figure 4.

The format of the new MULTI_FLOWSPEC object is given below:

```
       31           24 23           16 15            8 7             0
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  1   | 0 (a) |    reserved           |            19 (b)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  2   |    5  (c)     |0| reserved    |            18 (d)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  3   |   125 (e)     |    0 (f)      |             5 (g)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  4   |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  5   |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  6   |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  7   |  Minimum Policed Unit [m] (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  8   |  Maximum Packet Size [M]  (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  9   |   125 (e)     |    0 (f)      |             5 (g)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 10   |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 11   |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 12   |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 13   |  Minimum Policed Unit [m] (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 14   |  Maximum Packet Size [M]  (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 15   |   125 (e)     |    0 (f)      |             5 (g)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 16   |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 17   |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 18   |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 19   |  Minimum Policed Unit [m] (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 20   |  Maximum Packet Size [M]  (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5. Multiple FLOWSPEC for Controlled-Load service

(a) - Message format version number (0)
(b) - Overall length (19 words not including header)
(c) - Service header, service number 5 (Controlled-Load)
(d) - Length of controlled-load data, 18 words not including per-service header
(e) - Parameter ID, parameter 125 (Multiple Token Bucket TSpec)
(f) - Parameter 125 flags (none set)
(g) - Parameter 125 length, 5 words not including per-service header

This is for the 2nd through Nth TSPEC in the RESV, in the preferred order.

The message format (a) remains the same for a second TSPEC and for additional TSPECs.

The Overall Length (b) includes the TSPECs, plus the 2nd Word (fields (c) and (d)), which MUST NOT be repeated. The service header fields (e),(f) and(g), which are repeated for each TSPEC.

The Service header, here service number 5 (Controlled-Load) MUST remain the same for the RESV message. The services, Controlled-Load and Guaranteed MUST NOT be mixed within the same RESV message. In other words, if one TSPEC is a Controlled Load service TSPEC, the remaining TSPECs MUST be Controlled Load service. This same rule also is true for Guaranteed Service - if one TSPEC is for Guaranteed

Service, the rest of the TSPECs in this PATH or RESV MUST be for Guaranteed Service.

The Length of controlled-load data (d) also increases to account for the additional TSPECs.

Each FLOWSPEC is six 32-bit Words long (the per-service header plus the 5 values that are 1 Word each in length), therefore the length is in 6 Word increments for each additional TSPEC. Case in point, from the above Figure 5, Words 3-8 are the first TSPEC (2nd preferred), Words 9-14 are the next TSPEC (3rd preferred), and Words 15-20 are the final TSPEC (and 4th preferred) in this example of 3 TSPECs in this FLOWSPEC. There is no limit placed on the number of TSPECs a particular FLOWSPEC can have.

Within the MULTI_FLOWSPEC, any SENDER_TSPEC that cannot be reserved - based on the information gathered in the ADSPEC, is not placed in the RESV or based on other information available to the receiver. Otherwise, the order in which the TSPECs were in the PATH message MUST be in the same order they are in the FLOWSPEC in the RESV. This is the order of preference of the sender, and MUST be maintained throughout the reservation establishment, unless the ADSPEC indicates one or more TSPECs cannot be granted, or the receiver cannot include any TSPEC due to technical or administrative constraints or one or more routers along the RESV path cannot grant a particular TSPEC. At any router that a reservation cannot honor a TSPEC, this TSPEC MUST be removed from the RESV, or else another router along the RESV path might reserve that TSPEC. This rule ensures this cannot happen.

Once one TSPEC has been removed from the RESV, the next in line TSPEC becomes the preferred TSPEC for that reservation. That router MUST generate a ResvErr message, containing an ERROR_SPEC object with a Policy Control Failure with Error code = 2 (Policy Control Failure), and an Error Value sub-code 102 (ERR_PARTIAL_PREEMPT) to the previous routers, clearing the now over allocation of bandwidth for this reservation. The difference between the previously accepted TSPEC bandwidth and the currently accepted TSPEC bandwidth is the amount this error identifies as the amount of bandwidth that is no longer required to be reserved. The ResvErr and the RESV messages are independent, and not normally sent by the same router. This aspect of this document is the extension to RFC 2205 (RSVP).

If a RESV cannot grant the final TSPEC, normal RSVP rules apply with regard to the transmission of a particular ResvErr.

3.4 Multiple FLOWSPEC for Guaranteed service

The FLOWSPEC object, which is used to request guaranteed service contains a TSPEC and RSpec. Here is the FLOWSPEC object from [RFC2215] when requesting Guaranteed service:

```
     31           24 23           16 15            8 7             0
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  1  | 0 (a) |    Unused             |            10 (b)             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  2  |    2  (c)     |0| reserved    |             9 (d)             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  3  |   127 (e)     |    0 (f)      |             5 (g)             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  4  |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  5  |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  6  |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  7  |  Minimum Policed Unit [m] (32-bit integer)                    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  8  |  Maximum Packet Size [M]  (32-bit integer)                    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  9  |     130 (h)   |    0 (i)      |            2 (j)              |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  10 |  Rate [R]  (32-bit IEEE floating point number)                |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  11 |  Slack Term [S]  (32-bit integer)                             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6. FLOWSPEC for Guaranteed service

(a) - Message format version number (0)
(b) - Overall length (9 words not including header)
(c) - Service header, service number 2 (Guaranteed)
(d) - Length of per-service data, 9 words not including per-service header
(e) - Parameter ID, parameter 127 (Token Bucket TSpec)
(f) - Parameter 127 flags (none set)
(g) - Parameter 127 length, 5 words not including parameter header
(h) - Parameter ID, parameter 130 (Guaranteed Service RSpec)
(i) - Parameter xxx flags (none set)
(j) - Parameter xxx length, 2 words not including parameter header

The difference in structure between the Controlled-Load FLOWSPEC and Guaranteed FLOWSPEC is the RSPEC, defined in [RFC2212].

For completeness, Figure 6 is included in its original form for backwards compatibility reasons, as if there were only 1 FLOWSPEC in the RESV. What is new when there is more than one TSPEC in the FLOWSPEC in a RESV message is the new MULTI_FLOWSPEC object in Figure 7 containing, for example, 3 FLOWSPECs requesting Guaranteed Service.

```
       31           24 23           16 15            8 7             0
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  1   | 0 (a) |    Unused             |            28 (b)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  2   |    2  (c)     |0| reserved    |            27 (d)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  3   |   125 (e)     |    0 (f)      |             5 (g)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  4   |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  5   |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  6   |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  7   |  Minimum Policed Unit [m] (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  8   |  Maximum Packet Size [M]  (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  9   |     124 (h)   |    0 (i)      |            2 (j)              |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  10  |  Rate [R]  (32-bit IEEE floating point number)                |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  11  |  Slack Term [S]  (32-bit integer)                             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  12  |   125 (e)     |    0 (f)      |             5 (g)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  13  |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  14  |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  15  |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  16  |  Minimum Policed Unit [m] (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  17  |  Maximum Packet Size [M]  (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  18  |     124 (h)   |    0 (i)      |            2 (j)              |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  19  |  Rate [R]  (32-bit IEEE floating point number)                |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  20  |  Slack Term [S]  (32-bit integer)                             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  21  |   125 (e)     |    0 (f)      |             5 (g)             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  22  |  Token Bucket Rate [r] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  23  |  Token Bucket Size [b] (32-bit IEEE floating point number)    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  24  |  Peak Data Rate [p] (32-bit IEEE floating point number)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  25  |  Minimum Policed Unit [m] (32-bit integer)                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 26  |  Maximum Packet Size [M]  (32-bit integer)                    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 27  |     124 (h)   |    0 (i)      |            2 (j)              |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 28  |  Rate [R]  (32-bit IEEE floating point number)                |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 29  |  Slack Term [S]  (32-bit integer)                             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7. Multiple FLOWSPECs for Guaranteed service

(a) - Message format version number (0)
(b) - Overall length (9 words not including header)
(c) - Service header, service number 2 (Guaranteed)
(d) - Length of per-service data, 9 words not including per-service header
(e) - Parameter ID, parameter 125 (Token Bucket TSpec)
(f) - Parameter 125 flags (none set)
(g) - Parameter 125 length, 5 words not including parameter header
(h) - Parameter ID, parameter 124 (Guaranteed Service RSpec)
(i) - Parameter 124 flags (none set)
(j) - Parameter 124 length, 2 words not including parameter header

There MUST be 1 RSPEC per TSPEC for Guaranteed Service. Therefore, there are 5 words for Receiver TSPEC and 3 words for the RSPEC. Therefore, for Guaranteed Service, the TSPEC/RSPEC combination occurs in increments of 8 words.

4. Rules of Usage

The following rules apply to nodes adhering to this specification:

4.1 Backward Compatibility

If the recipient does not understand this extension, it ignores this MULTI_TSPEC object, and operates normally for a node receiving this RSVP message.

4.2 Applies to Only a Single Session

When there is more than one TSPEC object or more than one FLOWSPEC object, this MUST NOT be considered for more than one flow created. These are OR choices for the same flow of data. In order to attain three reservations between two endpoints, three different reservation requests are required, not one reservation request with 3 TSPECs.

4.3 No Special Error Handling for PATH Message

If a problem occurs with the PATH message - regardless of this extension, normal RSVP procedures apply (i.e., there is no new PathErr code created within this extension document) - resulting in a PathErr message being sent upstream towards the sender, as usual.

4.4 Preference Order to be Maintained

When more than one TSPEC is in a PATH message, the order of TSPECs is decided by the Sender and MUST be maintained within the SENDER_TSPEC. The same order MUST be carried to the FLOWSPECs by the receiver. No additional TSPECS can be introduced by the receiver or any router processing these new objects. The deletion of TSPECs from a PATH message is not permitted. The deletion of the TSPECs when forming the FLOWSPEC is allowed by the receiver in the following cases:

- If one or more preferred TSPECs cannot be granted by a router as discovered during processing of the ADSPEC by the receiver, then they can be omitted when creating the FLOWSPEC(s) from the TSPECs.

- If one or more TSPECs arriving from the sender is not preferred by the receiver, then the receiver MAY omit any while creating the FLOWSPEC. A good reason to omit a TSPEC is if, for example, it does not match a codec supported by the receiver's application(s).

The deletion of the TSPECs in the router during the processing of this MULTI_FLOWSPEC object is allowed in the following cases:

- If the original FLOWSPEC cannot be granted by a router then the router may discard that FLOWSPEC and replace it with the topmost FLOWSPEC from the MULTI_FLOWSPEC project. This will cause the topmost FLOWSPEC in the MULTI_FLOWSPEC object to be removed. The next FLOWSPECs becomes the topmost FLOWSPEC.

- If the router merges multiple RESV into a single RESV message, then the FLOWSPEC and the multiple FLOWSPEC may be affected

The preferred order of the remaining TSPECs or FLOWSPECs MUST be kept intact both at the receiver as well as the router processing these objects.

4.5 Bandwidth Reduction in Downstream Routers

If there are multiple FLOWSPECs in a single RESV message, it is quite possible that a higher bandwidth is reserved at a previous downstream device. Thus, any device that grants a reservation that is not the highest will have to inform the previous downstream routers to reduce the bandwidth reserved for this particular

session.

The bandwidth reduction RFC [RFC4495] does not address the need that this document addresses. RFC 4495 defines an ability to preempt part of an existing reservation so as to admit a new incoming reservation with a higher priority, in lieu of tearing down the whole reservation having a lower priority. It does not specify the capability to reduce the bandwidth a RESV set up along the data path before the reservation is realized (from source to destination), when a subsequent router cannot support a more preferred FLOWSPEC contained in that RESV. This document extends the RFC 4495 defined partial teardown error to reduce bandwidth from previous downstream hops while a reservation is being established.

For example, if a 12Mbps TSPEC were granted for a reservation on previous hops, but could not be granted at the current hop, while the 4Mbps TSPEC could be granted (provided there is a MULTI_TSPEC with a 4Mbps TSPEC), this modification to the bandwidth reduction function would work by having the 4Mbps granting node send a reduction error to the downstream routers that installed 12Mbps for this reservation, thus clearing bandwidth that is now unnecessarily installed for a 4Mbps reservation.

4.6 Merging Rules

RFC 2205 defines the rules for merging as combining more than one FLOWSPEC into a single FLOWSPEC. In the case of MULTI_FLOWSPECs, merging of the two (or more) MULTI_FLOWSPEC MUST be done to arrive at a single MULTI_FLOWSPEC. The merged MULTI_FLOWSPEC will contain all the flow specification components of the individual MULTI_FLOWSPECs in descending orders of bandwidth. In other words, the merged FLOWSPEC MUST maintain the relative order of each of the individual FLOWSPECs. For example, if the individual FLOWSPEC order is 1,2,3 and another FLOWSPEC is a,b,c, then this relative ordering cannot be altered in the merged FLOWSPEC.

A byproduct of this is the ordering between the two individual FLOWSPECs cannot be signaled with this extension. If two (or more) FLOWSPECs have the same bandwidth, they are to be merged into one FLOWSPEC using the rules defined in RFC 2205. It is RECOMMENDED that the following rules are used for determining ordering (in TSPEC and FLOWSPEC):

For Controlled Load - in descending order of BW based on the Token Bucket Rate 'r' parameter value

For Guaranteed Service - in descending order of BW based on the RSPEC Rate 'R' parameter value

The resultant FLOWSPEC is added to the MULTI_FLOWSPEC based on its bandwidth in descending orders of bandwidth.

As a result of such merging, the number of FLOWSPECs in a MULTI_FLOWSPEC object should be the sum of the number of FLOWSPECs from individual MULTI_FLOWSPEC that have been merged *minus* the number of duplicates.

4.7 Applicability to Multicast

An RSVP message with a MULTI_TSPEC works just as well in a multicast scenario as it does in a unicast scenario. In a multicast scenario, the bandwidth allotted in each hop is the lowest bandwidth that can be admitted along the various path. For example:

```
+--------+       +----------+      +----------+      +------------+
| sender |======>| Router-1 |=====>| Router-2 |=====>| Receiver-A |
+--------+       +----------+      +----------+      +------------+
                      |                 |
                      |                 |
                      |                 V
                      |           +------------+
                      |           | Receiver-C |
                      |           +------------+
                      |
                      V
                +------------+
                | Receiver-B |
                +------------+
```

Figure 8. MULTI_TPSEC and Multicast

If the sender (in Figure 8) sends 3 TSPECs (i.e., 1 TSPEC Object, and 2 in the MULTI_TSPEC Object) of 12Mbps, 5Mbps and 1.5Mbps. Let us say the path from Receiver-B to Router-1 admitted 5Mbps, Receiver-C to Router-2 admitted 1.5Mbps and Receiver-A to Router-2 admitted 12Mbps.

When the Resv message is send upstream from Router-2, the combining of 1.5Mbps (to Receiver-C) and 12Mbps (to Receiver-A) will be resolved to 1.5Mbps (lowest that can be admitted). Only a Resv with 1.5Mbps will be sent upstream from Router-2. Likewise, at Router-1, the combining of 1.5Mbps (to Router-2) and 5Mbps (to Receiver-B) will be resolved to 1.5Mbps units.

This is to allow the sender to transmit the flow at a rate that can be accepted by all devices along the path. Without this, if Router-2 receives a flow of 12Mbps, it will not know how to create a flow of 1.5Mbps down to Receiver-B. A differentiated reservation for the various paths along a multicast path is only possible with a Media-aware network device (MANE). The discussion of MANE and how it relates to admission control is outside the scope of this draft.

4.8 MULTI_TSPEC Specific Error

Since this mechanism is backward compatible, it is possible that a router without support for this MULTI_TSPEC extension will reject a reservation because the bandwidth indicated in the primary FLOWSPECs is not available. This means that an attempt with a lower bandwidth might have been successful, if one were included in a MULTI_TSPEC Object. Therefore, one should be able to differentiate between an admission control error where there is insufficient bandwidth when all the FLOWSPECs are considered and insufficient bandwidth when only the primary FLOWSPEC is considered.

This requires the definition of an error code within the ERROR_SPEC Object. When a router does not have sufficient bandwidth even after considering all the FLOWSPEC provided, it issues a new "MULTI_TSPEC bandwidth unavailable " error. This will be an Admission Control Failure (error #1), with a subcode of 6. A router that does not support this MULTI_TSPEC extension will return the "requested bandwidth unavailable" error as defined in RFC 2205 as if there was no MULTI_TSPEC in the message.

4.9 Other Considerations

- RFC 4495 articulates why a ResvErr is more appropriate to use for reducing the bandwidth of an existing reservation vs. a ResvTear.

- Refreshes only include the TSPECs that were accepted. One SHOULD be sent immediately upon the Sender receiving the RESV, to ensure all routers in this flow are synchronized with which TSPEC is in place.

- Periodically, it might be appropriate to attempt to increase the bandwidth of an accepted reservation with one of the TSPECs that were not accepted by the network when the reservation was first installed. This SHOULD NOT occur too regularly. This document currently offers no guidance on the frequency of this bump request for a rejected TSPEC from the PATH.

4.10 Known Open Issues

Here are the know open issues within this document:

o Need to ensure the cap on the number of TSPECs and FLOWSPECs is viable, yet controlled.

5. Security considerations

The security considerations for this document do not exceed what is already in RFC 2205 (RESV) or RFC 2210 (IntServ), as nothing in either of those documents prevent a node from requesting a lot of bandwidth in a single TSPEC. This document merely reduces the signaling traffic load on the network by allowing many requests that fall under the same policy controls to be included in a single round-trip message exchange.

Further, this document does not increase the security risk(s) to that defined in RFC 4495, where this document creates additional meaning to the RFC 4495 created error code 102.

A misbehaving Sender can include too many TSPECs in the MULTI_TSPEC object, which can lead to an amplification attack. That said, a bad implementation can create a reservation for each TSPEC received from within the Resv message. The number of TSPECs in the new MULTI_TSPEC object is limited, and the spec clearly states that only a single reservation is to be set up per Resv message.

To ensure the integrity of RSVP, the RSVP Authentication mechanisms defined in [RFC2747] and [RFC3097] SHOULD be used. Those protect RSVP message integrity hop-by-hop and provide node authentication as well as replay protection, thereby protecting against corruption and spoofing of RSVP messages.

6. IANA considerations

This document IANA registers the following new parameter name in the Integ-serv assignments at [IANA]:

Registry Name: Parameter Names
Registry:

Value	Description	Reference
125	Multiple_Token_Bucket_Tspec	[RFCXXXX]
124	Multiple_Guaranteed_Service_RSpec	[RFCXXXX]

Where RFCXXXX is replaced with the RFC number assigned to this Document.

This document IANA registers the following new error subcode in the Error code section, under the Admission Control Failure (error=1), of the rsvp-parameters assignments at [IANA]:

Registry Name: Error Codes and Globally-Defined Error Value Sub-Codes
Registry:
"Admission Control Failure"

Error Subcode		meaning	Reference
6	=	MULTI_TSPEC bandwidth unavailable	[RFCXXXX]

7. Acknowledgments

The authors wish to thank Fred Baker, Joe Touch, Bruce Davie, Dave Oran, Ashok Narayanan, Lou Berger, Lars Eggert, Arun Kudur, Ken Carlberg and Janet Gunn for their helpful comments and guidance in this effort.

And to Francois Le Faucheur, who provided text in this version.

8. References

8.1. Normative References

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997

[RFC2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997

[RFC2210] J. Wroclawski, "The Use of RSVP with IETF Integrated Services", RFC 2210, September 1997

[RFC2212] S. Shenker, C. Partridge, R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, September 1997

[RFC2215] S. Shenker, J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements", RFC 2212, September 1997

[RFC2747] F. Baker, B. Lindell, M. Talwar, " RSVP Cryptographic Authentication", RFC2747, January 2000

[RFC3097] R. Braden, L. Zhang, "RSVP Cryptographic Authentication -- Updated Message Type Value", RFC 3097, April 2001

[RFC4495] J. Polk, S. Dhesikan, "A Resource Reservation Protocol (RSVP) Extension for the Reduction of Bandwidth of a Reservation Flow", RFC 4495, May 2006

8.2. Informative References

[IANA] http://www.iana.org/assignments/integ-serv

Authors' Addresses

James Polk
3913 Treemont Circle
Colleyville, Texas, USA
+1.817.271.3552

mailto: jmpolk@cisco.com

Subha Dhesikan
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134 USA

mailto: sdhesika@cisco.com

Appendix A: Alternatives for Sending Multiple TSPECs

This appendix describes the discussion within the TSVWG of which approach best fits the requirements of sending multiple TSPECs within a single PATH or RESV message. There were 3 different options proposed, of which - 2 were insufficient or caused more harm than other options.

Looking at the format of a PATH message [RFC2205] again:

```
        <PATH Message> ::= <Common Header> [ <INTEGRITY> ]

                                <SESSION> <RSVP_HOP>

                                <TIME_VALUES>

                               [ <POLICY_DATA> ... ]

                               [ <sender descriptor> ]

        <sender descriptor> ::= <SENDER_TEMPLATE> <SENDER_TSPEC>
                                                   ^^^^^^^^^^^^
                               [ <ADSPEC> ]
```

For the PATH message, the focus of this document is with what to do with respect to the <SENDER_TSPEC> above, highlighted by the '^^^^' characters. No other object within the PATH message will be affected by this IntServ extension.

The ADSPEC is optional in IntServ; therefore it might or might not be in the RSVP PATH message. Presently, the SENDER_TSPEC is limited to one bandwidth associated with the session. This is changed in this extension to IntServ to multiple bandwidths for the same session. There are multiple options on how the additional bandwidths

may be added:

Option #1 - creating the ability to add one or more additional (and complete) SENDER_TSPECs,

or

Option #2 - create the ability for the one already allowed SENDER_TSPEC to carry more than one bandwidth amount for the same reservation.

or

Option #3 - create the ability for the existing SENDER_TSPEC to remain unchanged, but add an optional <MULTI_TSPEC> object to the <sender descriptor> such as this:

```
        <sender descriptor> ::= <SENDER_TEMPLATE> <SENDER_TSPEC>

                                     [ <ADSPEC> ]  [ <MULTI_TSPEC> ]
                                                     ^^^^^^^^^^^
```

Here is another way of looking at the option choices:

```
+--------------------+----------------------+----------------------+
|    Option#1        |      Option#2        |      Option#3        |
|                    |                      |                      |
|   +----------+     |  +---------------+   |    +----------+      |
|   |  TSPEC1  |     |  |  MULTI_TSPEC  |   |    |  TSPEC1  |      |
|   +----------+     |  |    Object     |   |    +----------+      |
|                    |  | +--------+    |   |                      |
|   +----------+     |  | | TSPEC1 |    |   |  +---------------+   |
|   |  TSPEC2  |     |  | +--------+    |   |  |  MULTI_TSPEC  |   |
|   +----------+     |  | +--------+    |   |  |    Object     |   |
|                    |  | | TSPEC2 |    |   |  | +--------+    |   |
|   +----------+     |  | +--------+    |   |  | | TSPEC2 |    |   |
|   |  TSPEC3  |     |  | +--------+    |   |  | +--------+    |   |
|   +----------+     |  | | TSPEC3 |    |   |  | +--------+    |   |
|                    |  | +--------+    |   |  | | TSPEC3 |    |   |
|   +----------+     |  | | TSPEC4 |    |   |  | +--------+    |   |
|   |  TSPEC4  |     |  | +--------+    |   |  | +--------+    |   |
|   +----------+     |  +---------------+   |  | | TSPEC4 |    |   |
|                    |                      |  | +--------+    |   |
|                    |                      |  +---------------+   |
|                    |                      |                      |
+--------------------+----------------------+----------------------+
```

Figure 3. Concept of Option Choice

Option #1 and #2 do not allow for backward compatibility. If the currently used SENDER_TSPEC and FLOWSPEC objects are changed, then unless all the routers requiring RSVP processing are upgraded, this

functionality cannot be realized. As it is unlikely that all routers along the path will have the necessary enhancements as per this extension at one given time, therefore, it is necessary this enhancement be made in a way that is backward compatible. Therefore, option #1 and option #2 has been discarded in favor of option #3, which had WG consensus in a recent IETF meeting.

Option #3: This option has the advantage of being backwards compatible with existing implementations of [RFC2205] and [RFC2210], as the multiple TSPECs and FLOWSPECs are inserted as optional objects and such objects do not need to be processed, especially if they are not understood.

Option #3 applies to the FLOWSPEC contained in the RESV message as well. In this option, the original SENDER_TSPEC and the FLOWSPEC are left untouched, allowing routers not supporting this extension to be able to process the PATH and the RESV message without issue. Two new additional objects are defined in this document. They are the MULTI_TSPEC and the MULTI_FLOWSPEC for the PATH and the RESV message, respectively. The additional TSPECs (in the new MULTI_TSPEC Object) are included in the PATH and the additional FLOWSPECS (in the new MULTI_FLOWSPEC Object) are included in the RESV message as new (optional) objects. These additional objects will have a class number of 11bbbbbb, allowing older routers to ignore the object(s) and forward each unexamined and unchanged, as defined in section 3.10 of [RFC 2205].

We state in the document body that the top most FLOWSPEC of the new MULTI_FLOWSPEC Object in the RESV message replaces the existing FLOWSPEC when it is determined by the receiver (perhaps along with the ADSPEC) that the original FLOWSPEC cannot be granted. Therefore, the ordering of preference issue is solved with Option#3 as well.

NOTE: it is important to emphasize here that including more than one FLOWSPEC in the RESV message does not cause more than one FLOWSPEC to be granted. This document requires that the receiver arrange these multiple FLOWSPECs in the order of preference according to the order remaining from the MULTI_TSPECs in the PATH message. The benefit of this arrangement is that RSVP does not have to process the rest of the FLOWSPEC if it can admit the first one.

Additional details of these options can be found in the draft-polk-tsvwg-...-01 version of this appendix (which includes the RSVP bit mapping of fields in the TSPECs, if the reader wishes to search for that doc.

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 April 2022

R. R. Stewart
Netflix, Inc.
M. Tüxen
I. Rüngeler
Münster Univ. of Appl. Sciences

25 October 2021

Stream Control Transmission Protocol (SCTP) Network Address Translation Support

draft-ietf-tsvwg-natsupp-23

Abstract

The Stream Control Transmission Protocol (SCTP) provides a reliable communications channel between two end-hosts in many ways similar to the Transmission Control Protocol (TCP). With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT functions for TCP that allows multiple hosts to reside behind a NAT function and yet share a single IPv4 address, even when two hosts (behind a NAT function) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation (NAPT).

This document describes the protocol extensions needed for the SCTP endpoints and the mechanisms for NAT functions necessary to provide similar features of NAPT in the single point and multipoint traversal scenario.

Finally, a YANG module for SCTP NAT is defined.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Stream Control Transmission Protocol (SCTP) [RFC4960] provides a reliable communications channel between two end-hosts in many ways similar to TCP [RFC0793]. With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT functions for TCP that allows multiple hosts to reside behind a NAT function using private-use addresses (see [RFC6890]) and yet share a single IPv4 address, even when two hosts (behind a NAT function) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation (NAPT). Please note that this document focuses on the case where the NAT function maps a single or multiple internal addresses to a single external address and vice versa.

To date, specialized code for SCTP has not yet been added to most NAT functions so that only a translation of IP addresses is supported. The end result of this is that only one SCTP-capable host can successfully operate behind such a NAT function and this host can only be single-homed. The only alternative for supporting legacy NAT functions is to use UDP encapsulation as specified in [RFC6951].

The NAT function in the document refers to NAPT functions described in Section 2.2 of [RFC3022], NAT64 [RFC6146], or DS-Lite AFTR [RFC6333].

This document specifies procedures allowing a NAT function to support SCTP by providing similar features to those provided by a NAPT for TCP (see [RFC5382] and [RFC7857]), UDP (see [RFC4787] and [RFC7857]), and ICMP (see [RFC5508] and [RFC7857]). This document also specifies a set of data formats for SCTP packets and a set of SCTP endpoint procedures to support NAT traversal. An SCTP implementation supporting these procedures can assure that in both single-homed and multi-homed cases a NAT function will maintain the appropriate state without the NAT function needing to change port numbers.

It is possible and desirable to make these changes for a number of reasons:

* It is desirable for SCTP internal end-hosts on multiple platforms to be able to share a NAT function's external IP address in the same way that a TCP session can use a NAT function.

* If a NAT function does not need to change any data within an SCTP packet, it will reduce the processing burden of NAT'ing SCTP by not needing to execute the CRC32c checksum used by SCTP.

* Not having to touch the IP payload makes the processing of ICMP messages by NAT functions easier.

An SCTP-aware NAT function will need to follow these procedures for generating appropriate SCTP packet formats.

When considering SCTP-aware NAT it is possible to have multiple levels of support. At each level, the Internal Host, Remote Host, and NAT function does or does not support the procedures described in this document. The following table illustrates the results of the various combinations of support and if communications can occur between two endpoints.

Internal Host	NAT Function	Remote Host	Communication
Support	Support	Support	Yes
Support	Support	No Support	Limited
Support	No Support	Support	None
Support	No Support	No Support	None
No Support	Support	Support	Limited
No Support	Support	No Support	Limited
No Support	No Support	Support	None
No Support	No Support	No Support	None

Table 1: Communication possibilities

From the table it can be seen that no communication can occur when a NAT function does not support SCTP-aware NAT. This assumes that the NAT function does not handle SCTP packets at all and all SCTP packets sent from behind a NAT function are discarded by the NAT function. In some cases, where the NAT function supports SCTP-aware NAT, but one of the two hosts does not support the feature, communication can possibly occur in a limited way. For example, only one host can have a connection when a collision case occurs.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

This document uses the following terms, which are depicted in Figure 1. Familiarity with the terminology used in [RFC4960] and [RFC5061] is assumed.

Internal-Address (Int-Addr)
An internal address that is known to the internal host.

Internal-Port (Int-Port)
The port number that is in use by the host holding the Internal-Address.

Internal-VTag (Int-VTag)
The SCTP Verification Tag (VTag) (see Section 3.1 of [RFC4960]) that the internal host has chosen for an association. The VTag is a unique 32-bit tag that accompanies any incoming SCTP packet for this association to the Internal-Address.

Remote-Address (Rem-Addr)
The address that an internal host is attempting to contact.

Remote-Port (Rem-Port)
The port number used by the host holding the Remote-Address.

Remote-VTag (Rem-VTag)
The Verification Tag (VTag) (see Section 3.1 of [RFC4960]) that the host holding the Remote-Address has chosen for an association. The VTag is a unique 32-bit tag that accompanies any outgoing SCTP packet for this association to the Remote-Address.

External-Address (Ext-Addr)
An external address assigned to the NAT function, that it uses as a source address when sending packets towards a Remote-Address.

```
     Internal Network    |         External Network
                         |
               Internal  |   External              Remote
               Address   |   Address  /--\/--\    Address
  +--------+         +-----+        /        \           +--------+
  | Host A |=========| NAT |=======| Network  |==========| Host B |
  +--------+         +-----+        \        /           +--------+
               Internal  |            \--/\--/     Remote
    Internal     Port    |                           Port   Remote
      VTag               |                                   VTag
```

Figure 1: Basic Network Setup

4. Motivation and Overview

4.1. SCTP NAT Traversal Scenarios

This section defines the notion of single and multipoint NAT traversal.

4.1.1. Single Point Traversal

In this case, all packets in the SCTP association go through a single NAT function, as shown in Figure 2.

```
   Internal Network    |       External Network
                       |
                       |               /--\/--\
 +--------+         +-----+           /        \          +--------+
 | Host A |=========| NAT |========= | Network  | ========| Host B |
 +--------+         +-----+           \        /          +--------+
                       |               \--/\--/
                       |
```

Figure 2: Single NAT Function Scenario

A variation of this case is shown in Figure 3, i.e., multiple NAT functions in the forwarding path between two endpoints.

```
       Internal | External : Internal | External
                |          :          |
                |          :          |       /--\/--\
 +--------+  +-----+       :       +-----+   /        \   +--------+
 | Host A |==| NAT |=======:=======| NAT |==| Network  |==| Host B |
 +--------+  +-----+       :       +-----+   \        /   +--------+
                |          :          |       \--/\--/
                |          :          |
```

Figure 3: Serial NAT Functions Scenario

Although one of the main benefits of SCTP multi-homing is redundant paths, in the single point traversal scenario the NAT function represents a single point of failure in the path of the SCTP multi-homed association. However, the rest of the path can still benefit from path diversity provided by SCTP multi-homing.

The two SCTP endpoints in this case can be either single-homed or multi-homed. However, the important thing is that the NAT function in this case sees all the packets of the SCTP association.

4.1.2. Multipoint Traversal

This case involves multiple NAT functions and each NAT function only sees some of the packets in the SCTP association. An example is shown in Figure 4.

```
             Internal      |      External
                           +------+             /---\/---\
                /=======|NAT A |=========\  /           \
    +--------+ /          +------+          \/            \    +--------+
    | Host A |/             |               |    Network   |===| Host B |
    +--------+\             |               \            /    +--------+
               \           +------+          / \          /
                \=======|NAT B |=========/   \---\/---/
                           +------+
                              |
```

Figure 4: Parallel NAT Functions Scenario

This case does not apply to a single-homed SCTP association (i.e., both endpoints in the association use only one IP address). The advantage here is that the existence of multiple NAT traversal points can preserve the path diversity of a multi-homed association for the entire path. This in turn can improve the robustness of the communication.

4.2. Limitations of Classical NAPT for SCTP

Using classical NAPT possibly results in changing one of the SCTP port numbers during the processing, which requires the recomputation of the transport layer checksum by the NAPT function. Whereas for UDP and TCP this can be done very efficiently, for SCTP the checksum (CRC32c) over the entire packet needs to be recomputed (see Appendix B of [RFC4960] for details of the CRC32c computation). This would considerably add to the NAT computational burden, however hardware support can mitigate this in some implementations.

An SCTP endpoint can have multiple addresses but only has a single port number to use. To make multipoint traversal work, all the NAT functions involved need to recognize the packets they see as belonging to the same SCTP association and perform port number translation in a consistent way. One possible way of doing this is to use a pre-defined table of port numbers and addresses configured within each NAT function. Other mechanisms could make use of NAT to NAT communication. Such mechanisms have not been deployed on a wide scale base and thus are not a preferred solution. Therefore an SCTP variant of NAT function has been developed (see Section 4.3).

4.3. The SCTP-Specific Variant of NAT

In this section it is allowed that there are multiple SCTP capable hosts behind a NAT function that share one External-Address. Furthermore, this section focuses on the single point traversal scenario (see Section 4.1.1).

The modification of outgoing SCTP packets sent from an internal host is simple: the source address of the packets has to be replaced with the External-Address. It might also be necessary to establish some state in the NAT function to later handle incoming packets.

Typically, the NAT function has to maintain a NAT binding table of Internal-VTag, Internal-Port, Remote-VTag, Remote-Port, Internal-Address, and whether the restart procedure is disabled or not. An entry in that NAT binding table is called a NAT-State control block. The function Create() obtains the just mentioned parameters and returns a NAT-State control block. A NAT function MAY allow creating NAT-State control blocks via a management interface.

For SCTP packets coming from the external realm of the NAT function the destination address of the packets has to be replaced with the Internal-Address of the host to which the packet has to be delivered, if a NAT state entry is found. The lookup of the Internal-Address is based on the Remote-VTag, Remote-Port, Internal-VTag and the Internal-Port.

The entries in the NAT binding table need to fulfill some uniqueness conditions. There can not be more than one entry NAT binding table with the same pair of Internal-Port and Remote-Port. This rule can be relaxed, if all NAT binding table entries with the same Internal-Port and Remote-Port have the support for the restart procedure disabled (see Section 5.3.1). In this case there can not be no more than one entry with the same Internal-Port, Remote-Port and Remote-VTag and no more than one NAT binding table entry with the same Internal-Port, Remote-Port, and Int-VTag.

The processing of outgoing SCTP packets containing an INIT chunk is illustrated in the following figure. This scenario is valid for all message flows in this section.

```
                                          /--\/--\
   +--------+          +-----+           /        \           +--------+
   | Host A | <------> | NAT | <------> | Network  | <------> | Host B |
   +--------+          +-----+           \        /           +--------+
                                          \--/\---/


                INIT[Initiate-Tag]
    Int-Addr:Int-Port ------> Rem-Addr:Rem-Port
                     Rem-VTag=0

                Create(Initiate-Tag, Int-Port, 0, Rem-Port, Int-Addr,
                       IsRestartDisabled)
                Returns(NAT-State control block)

              Translate To:

                            INIT[Initiate-Tag]
              Ext-Addr:Int-Port ------> Rem-Addr:Rem-Port
                                Rem-VTag=0
```

Normally a NAT binding table entry will be created.

However, it is possible that there is already a NAT binding table entry with the same Remote-Port, Internal-Port, and Internal-VTag but different Internal-Address and the restart procedure is disabled. In this case the packet containing the INIT chunk MUST be dropped by the NAT and a packet containing an ABORT chunk SHOULD be sent to the SCTP host that originated the packet with the M bit set and 'VTag and Port Number Collision' error cause (see Section 5.1.1 for the format). The source address of the packet containing the ABORT chunk MUST be the destination address of the packet containing the INIT chunk.

If an outgoing SCTP packet contains an INIT or ASCONF chunk and a matching NAT binding table entry is found, the packet is processed as a normal outgoing packet.

It is also possible that a NAT binding table entry with the same Remote-Port and Internal-Port exists without an Internal-VTag conflict but there exists a NAT binding table entry with the same port numbers but a different Internal-Address and the restart procedure is not disabled. In such a case the packet containing the INIT chunk MUST be dropped by the NAT function and a packet containing an ABORT chunk SHOULD be sent to the SCTP host that originated the packet with the M bit set and 'Port Number Collision' error cause (see Section 5.1.1 for the format).

The processing of outgoing SCTP packets containing no INIT chunks is described in the following figure.

```
                                          /--\/--\
+--------+          +-----+            /        \            +--------+
| Host A | <------> | NAT | <------> | Network  | <------> | Host B |
+--------+          +-----+            \        /            +--------+
                                          \--/\---/

Int-Addr:Int-Port ------> Rem-Addr:Rem-Port
                  Rem-VTag


                         Translate To:

                         Ext-Addr:Int-Port ------> Rem-Addr:Rem-Port
                                           Rem-VTag
```

The processing of incoming SCTP packets containing an INIT ACK chunk is illustrated in the following figure. The Lookup() function has as input the Internal-VTag, Internal-Port, Remote-VTag, and Remote-Port. It returns the corresponding entry of the NAT binding table and updates the Remote-VTag by substituting it with the value of the Initiate-Tag of the INIT ACK chunk. The wildcard character signifies that the parameter's value is not considered in the Lookup() function or changed in the Update() function, respectively.

```
                                          /--\/--\
+--------+          +-----+            /        \            +--------+
| Host A | <------> | NAT | <------> | Network  | <------> | Host B |
+--------+          +-----+            \        /            +--------+
                                          \--/\---/

                                            INIT ACK[Initiate-Tag]
                            Ext-Addr:Int-Port <---- Rem-Addr:Rem-Port
                                              Int-VTag

         Lookup(Int-VTag, Int-Port, *, Rem-Port)
         Update(*, *, Initiate-Tag, *)

         Returns(NAT-State control block containing Int-Addr)

               INIT ACK[Initiate-Tag]
 Int-Addr:Int-Port <------ Rem-Addr:Rem-Port
                   Int-VTag
```

In the case where the Lookup function fails because it does not find an entry, the SCTP packet is dropped. If it succeeds, the Update routine inserts the Remote-VTag (the Initiate-Tag of the INIT ACK chunk) in the NAT-State control block.

The processing of incoming SCTP packets containing an ABORT or SHUTDOWN COMPLETE chunk with the T bit set is illustrated in the following figure.

```
                                        /--\/--\
+--------+          +-----+           /        \           +--------+
| Host A | <------> | NAT | <------> | Network  | <------> | Host B |
+--------+          +-----+           \        /           +--------+
                                        \--/\---/

                          Ext-Addr:Int-Port <------ Rem-Addr:Rem-Port
                                                    Rem-VTag

         Lookup(*, Int-Port, Rem-VTag, Rem-Port)

         Returns(NAT-State control block containing Int-Addr)

 Int-Addr:Int-Port <------ Rem-Addr:Rem-Port
                   Rem-VTag
```

For an incoming packet containing an INIT chunk a table lookup is made only based on the addresses and port numbers. If an entry with a Remote-VTag of zero is found, it is considered a match and the Remote-VTag is updated. If an entry with a non-matching Remote-VTag is found or no entry is found, the incoming packet is silently dropped. If an entry with a matching Remote-VTag is found, the incoming packet is forwarded. This allows the handling of INIT collision through NAT functions.

The processing of other incoming SCTP packets is described in the following figure.

```
                                          /--\/--\
   +--------+          +-----+           /        \            +--------+
   | Host A | <------> | NAT | <------> | Network  | <------> | Host B |
   +--------+          +-----+           \        /            +--------+
                                          \--/\---/

                            Ext-Addr:Int-Port <------ Rem-Addr:Rem-Port
                                              Int-VTag

            Lookup(Int-VTag, Int-Port, *, Rem-Port)

            Returns(NAT-State control block containing Internal-Address)

    Int-Addr:Int-Port <------ Rem-Addr:Rem-Port
                      Int-VTag
```

5. Data Formats

This section defines the formats used to support NAT traversal. Section 5.1 and Section 5.2 describe chunks and error causes sent by NAT functions and received by SCTP endpoints. Section 5.3 describes parameters sent by SCTP endpoints and used by NAT functions and SCTP endpoints.

5.1. Modified Chunks

This section presents existing chunks defined in [RFC4960] for which additional flags are specified by this document.

5.1.1. Extended ABORT Chunk

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Type = 6    | Reserved  |M|T|           Length              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   \                                                               \
   /                   zero or more Error Causes                   /
   \                                                               \
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The ABORT chunk is extended to add the new 'M bit'. The M bit indicates to the receiver of the ABORT chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box (e.g., NAT).

[NOTE to RFC-Editor: Assignment of M bit to be confirmed by IANA.]

5.1.2. Extended ERROR Chunk

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 9    | Reserved  |M|T|           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                   zero or more Error Causes                   /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The ERROR chunk defined in [RFC4960] is extended to add the new 'M bit'. The M bit indicates to the receiver of the ERROR chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box.

[NOTE to RFC-Editor: Assignment of M bit to be confirmed by IANA.]

5.2. New Error Causes

This section defines the new error causes added by this document.

5.2.1. VTag and Port Number Collision Error Cause

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Cause Code = 0x00B0        |     Cause Length = Variable   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                             Chunk                             /
/                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the 'VTag and Port Number Collision' Error Cause. IANA is requested to assign the value 0x00B0 for this cause code.

Cause Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

Chunk: variable length

The Cause-Specific Information is filled with the chunk that caused this error. This can be an INIT, INIT ACK, or ASCONF chunk. Note that if the entire chunk will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

[NOTE to RFC-Editor: Assignment of cause code to be confirmed by IANA.]

5.2.2. Missing State Error Cause

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Cause Code = 0x00B1        |     Cause Length = Variable   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                       Original Packet                         /
/                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the 'Missing State' Error Cause. IANA is requested to assign the value 0x00B1 for this cause code.

Cause Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

Original Packet: variable length
The Cause-Specific Information is filled with the IPv4 or IPv6 packet that caused this error. The IPv4 or IPv6 header MUST be included. Note that if the packet will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

[NOTE to RFC-Editor: Assignment of cause code to be confirmed by IANA.]

5.2.3. Port Number Collision Error Cause

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Cause Code = 0x00B2        |     Cause Length = Variable   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                             Chunk                             /
/                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the 'Port Number Collision' Error Cause. IANA is requested to assign the value 0x00B2 for this cause code.

Cause Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

Chunk: variable length
The Cause-Specific Information is filled with the chunk that caused this error. This can be an INIT, INIT ACK, or ASCONF chunk. Note that if the entire chunk will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

[NOTE to RFC-Editor: Assignment of cause code to be confirmed by IANA.]

5.3. New Parameters

This section defines new parameters and their valid appearance defined by this document.

5.3.1. Disable Restart Parameter

This parameter is used to indicate that the restart procedure is requested to be disabled. Both endpoints of an association MUST include this parameter in the INIT chunk and INIT ACK chunk when establishing an association and MUST include it in the ASCONF chunk when adding an address to successfully disable the restart procedure.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Type = 0xC007         |         Length = 4            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Parameter Type: 2 bytes (unsigned integer)

This field holds the IANA defined parameter type for the Disable Restart Parameter. IANA is requested to assign the value 0xC007 for this parameter type.

Parameter Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the parameter. The value MUST be 4.

[NOTE to RFC-Editor: Assignment of parameter type to be confirmed by IANA.]

The Disable Restart Parameter MAY appear in INIT, INIT ACK and ASCONF chunks and MUST NOT appear in any other chunk.

5.3.2. VTags Parameter

This parameter is used to help a NAT function to recover from state loss.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Parameter Type = 0xC008   |     Parameter Length = 16     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ASCONF-Request Correlation ID                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Internal Verification Tag                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Remote Verification Tag                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Parameter Type: 2 bytes (unsigned integer)
This field holds the IANA defined parameter type for the VTags Parameter. IANA is requested to assign the value 0xC008 for this parameter type.

Parameter Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the parameter. The value MUST be 16.

ASCONF-Request Correlation ID: 4 bytes (unsigned integer)
This is an opaque integer assigned by the sender to identify each request parameter. The receiver of the ASCONF Chunk will copy this 32-bit value into the ASCONF Response Correlation ID field of the ASCONF ACK response parameter. The sender of the packet containing the ASCONF chunk can use this same value in the ASCONF ACK chunk to find which request the response is for. The receiver MUST NOT change the value of the ASCONF-Request Correlation ID.

Internal Verification Tag: 4 bytes (unsigned integer)
The Verification Tag that the internal host has chosen for the association. The Verification Tag is a unique 32-bit tag that accompanies any incoming SCTP packet for this association to the Internal-Address.

Remote Verification Tag: 4 bytes (unsigned integer)
The Verification Tag that the host holding the Remote-Address has chosen for the association. The VTag is a unique 32-bit tag that accompanies any outgoing SCTP packet for this association to the Remote-Address.

[NOTE to RFC-Editor: Assignment of parameter type to be confirmed by IANA.]

The VTags Parameter MAY appear in ASCONF chunks and MUST NOT appear in any other chunk.

6. Procedures for SCTP Endpoints and NAT Functions

If an SCTP endpoint is behind an SCTP-aware NAT, a number of problems can arise as it tries to communicate with its peers:

* IP addresses can not be included in the SCTP packet. This is discussed in Section 6.1.

* More than one host behind a NAT function could select the same VTag and source port number when communicating with the same peer server. This creates a situation where the NAT function will not be able to tell the two associations apart. This situation is discussed in Section 6.2.

* If an SCTP endpoint is a server communicating with multiple peers and the peers are behind the same NAT function, then the these peers cannot be distinguished by the server. This case is discussed in Section 6.3.

* A restart of a NAT function during a conversation could cause a loss of its state. This problem and its solution is discussed in Section 6.4.

* NAT functions need to deal with SCTP packets being fragmented at the IP layer. This is discussed in Section 6.5.

* An SCTP endpoint can be behind two NAT functions in parallel providing redundancy. The method to set up this scenario is discussed in Section 6.6.

The mechanisms to solve these problems require additional chunks and parameters, defined in this document, and modified handling procedures from those specified in [RFC4960] as described below.

6.1. Association Setup Considerations for Endpoints

The association setup procedure defined in [RFC4960] allows multi-homed SCTP endpoints to exchange its IP-addresses by using IPv4 or IPv6 address parameters in the INIT and INIT ACK chunks. However, this does not work when NAT functions are present.

Every association setup from a host behind a NAT function MUST NOT use multiple internal addresses. The INIT chunk MUST NOT contain an IPv4 Address parameter, IPv6 Address parameter, or Supported Address Types parameter. The INIT ACK chunk MUST NOT contain any IPv4 Address parameter or IPv6 Address parameter using non-global addresses. The INIT chunk and the INIT ACK chunk MUST NOT contain any Host Name parameters.

If the association is intended to be finally multi-homed, the procedure in Section 6.6 MUST be used.

The INIT and INIT ACK chunk SHOULD contain the Disable Restart parameter defined in Section 5.3.1.

6.2. Handling of Internal Port Number and Verification Tag Collisions

Consider the case where two hosts in the Internal-Address space want to set up an SCTP association with the same service provided by some remote hosts. This means that the Remote-Port is the same. If they both choose the same Internal-Port and Internal-VTag, the NAT function cannot distinguish between incoming packets anymore. However, this is unlikely. The Internal-VTags are chosen at random and if the Internal-Ports are also chosen from the ephemeral port range at random (see [RFC6056]) this gives a 46-bit random number that has to match.

The same can happen with the Remote-VTag when a packet containing an INIT ACK chunk or an ASCONF chunk is processed by the NAT function.

6.2.1. NAT Function Considerations

If the NAT function detects a collision of internal port numbers and verification tags, it SHOULD send a packet containing an ABORT chunk with the M bit set if the collision is triggered by a packet containing an INIT or INIT ACK chunk. If such a collision is triggered by a packet containing an ASCONF chunk, it SHOULD send a packet containing an ERROR chunk with the M bit. The M bit is a new

bit defined by this document to express to SCTP that the source of this packet is a "middle" box, not the peer SCTP endpoint (see Section 5.1.1). If a packet containing an INIT ACK chunk triggers the collision, the corresponding packet containing the ABORT chunk MUST contain the same source and destination address and port numbers as the packet containing the INIT ACK chunk. If a packet containing an INIT chunk or an ASCONF chunk, the source and destination address and port numbers MUST be swapped.

The sender of the packet containing an ERROR or ABORT chunk MUST include the error cause with cause code 'VTag and Port Number Collision' (see Section 5.2.1).

6.2.2. Endpoint Considerations

The sender of the packet containing the INIT chunk or the receiver of a packet containing the INIT ACK chunk, upon reception of a packet containing an ABORT chunk with M bit set and the appropriate error cause code for colliding NAT binding table state is included, SHOULD reinitiate the association setup procedure after choosing a new initiate tag, if the association is in COOKIE-WAIT state. In any other state, the SCTP endpoint MUST NOT respond.

The sender of the packet containing the ASCONF chunk, upon reception of a packet containing an ERROR chunk with M bit set, MUST stop adding the path to the association.

6.3. Handling of Internal Port Number Collisions

When two SCTP hosts are behind an SCTP-aware NAT it is possible that two SCTP hosts in the Internal-Address space will want to set up an SCTP association with the same server running on the same remote host. If the two hosts choose the same internal port, this is considered an internal port number collision.

For the NAT function, appropriate tracking can be performed by assuring that the VTags are unique between the two hosts.

6.3.1. NAT Function Considerations

The NAT function, when processing the packet containing the INIT ACK chunk, SHOULD note in its NAT binding table if the association supports the disable restart extension. This note is used when establishing future associations (i.e. when processing a packet containing an INIT chunk from an internal host) to decide if the connection can be allowed. The NAT function does the following when processing a packet containing an INIT chunk:

* If the packet containing the INIT chunk is originating from an internal port to a remote port for which the NAT function has no matching NAT binding table entry, it MUST allow the packet containing the INIT chunk creating an NAT binding table entry.

* If the packet containing the INIT chunk matches an existing NAT binding table entry, it MUST validate that the disable restart feature is supported and, if it does, allow the packet containing the INIT chunk to be forwarded.

* If the disable restart feature is not supported, the NAT function SHOULD send a packet containing an ABORT chunk with the M bit set.

The 'Port Number Collision' error cause (see Section 5.2.3) MUST be included in the ABORT chunk sent in response to the packet containing an INIT chunk.

If the collision is triggered by a packet containing an ASCONF chunk, a packet containing an ERROR chunk with the 'Port Number Collision' error cause SHOULD be sent in response to the packet containing the ASCONF chunk.

6.3.2. Endpoint Considerations

For the remote SCTP server this means that the Remote-Port and the Remote-Address are the same. If they both have chosen the same Internal-Port the server cannot distinguish between both associations based on the address and port numbers. For the server it looks like the association is being restarted. To overcome this limitation the client sends a Disable Restart parameter in the INIT chunk.

When the server receives this parameter it does the following:

* It MUST include a Disable Restart parameter in the INIT ACK to inform the client that it will support the feature.

* It MUST disable the restart procedures defined in [RFC4960] for this association.

Servers that support this feature will need to be capable of maintaining multiple connections to what appears to be the same peer (behind the NAT function) differentiated only by the VTags.

6.4. Handling of Missing State

6.4.1. NAT Function Considerations

If the NAT function receives a packet from the internal network for which the lookup procedure does not find an entry in the NAT binding table, a packet containing an ERROR chunk SHOULD be sent back with the M bit set. The source address of the packet containing the ERROR chunk MUST be the destination address of the packet received from the internal network. The verification tag is reflected and the T bit is set. Such a packet containing an ERROR chunk SHOULD NOT be sent if the received packet contains an ASCONF chunk with the VTags parameter or an ABORT, SHUTDOWN COMPLETE or INIT ACK chunk. A packet containing an ERROR chunk MUST NOT be sent if the received packet contains an ERROR chunk with the M bit set. In any case, the packet SHOULD NOT be forwarded to the remote address.

If the NAT function receives a packet from the internal network for which it has no NAT binding table entry and the packet contains an ASCONF chunk with the VTags parameter, the NAT function MUST update its NAT binding table according to the verification tags in the VTags parameter and, if present, the Disable Restart parameter.

When sending a packet containing an ERROR chunk, the error cause 'Missing State' (see Section 5.2.2) MUST be included and the M bit of the ERROR chunk MUST be set (see Section 5.1.2).

6.4.2. Endpoint Considerations

Upon reception of this packet containing the ERROR chunk by an SCTP endpoint the receiver takes the following actions:

* It SHOULD validate that the verification tag is reflected by looking at the VTag that would have been included in an outgoing packet. If the validation fails, discard the received packet containing the ERROR chunk.

* It SHOULD validate that the peer of the SCTP association supports the dynamic address extension. If the validation fails, discard the received packet containing the ERROR chunk.

* It SHOULD generate a packet containing a new ASCONF chunk containing the VTags parameter (see Section 5.3.2) and the Disable Restart parameter (see Section 5.3.1) if the association is using the disable restart feature. By processing this packet the NAT function can recover the appropriate state. The procedures for generating an ASCONF chunk can be found in [RFC5061].

The peer SCTP endpoint receiving such a packet containing an ASCONF chunk SHOULD add the address and respond with an acknowledgment if the address is new to the association (following all procedures defined in [RFC5061]). If the address is already part of the association, the SCTP endpoint MUST NOT respond with an error, but instead SHOULD respond with a packet containing an ASCONF ACK chunk acknowledging the address and take no action (since the address is already in the association).

Note that it is possible that upon receiving a packet containing an ASCONF chunk containing the VTags parameter the NAT function will realize that it has an 'Internal Port Number and Verification Tag collision'. In such a case the NAT function SHOULD send a packet containing an ERROR chunk with the error cause code set to 'VTag and Port Number Collision' (see Section 5.2.1).

If an SCTP endpoint receives a packet containing an ERROR chunk with 'Internal Port Number and Verification Tag collision' as the error cause and the packet in the Error Chunk contains an ASCONF with the VTags parameter, careful examination of the association is necessary. The endpoint does the following:

* It MUST validate that the verification tag is reflected by looking at the VTag that would have been included in the outgoing packet. If the validation fails, it MUST discard the packet.

* It MUST validate that the peer of the SCTP association supports the dynamic address extension. If the peer does not support this extension, it MUST discard the received packet containing the ERROR chunk.

* If the association is attempting to add an address (i.e. following the procedures in Section 6.6) then the endpoint MUST NOT consider the address part of the association and SHOULD make no further attempt to add the address (i.e. cancel any ASCONF timers and remove any record of the path), since the NAT function has a VTag collision and the association cannot easily create a new VTag (as it would if the error occurred when sending a packet containing an INIT chunk).

* If the endpoint has no other path, i.e. the procedure was executed due to missing a state in the NAT function, then the endpoint MUST abort the association. This would occur only if the local NAT function restarted and accepted a new association before attempting to repair the missing state (Note that this is no different than what happens to all TCP connections when a NAT function looses its state).

6.5. Handling of Fragmented SCTP Packets by NAT Functions

SCTP minimizes the use of IP-level fragmentation. However, it can happen that using IP-level fragmentation is needed to continue an SCTP association. For example, if the path MTU is reduced and there are still some DATA chunk in flight, which require packets larger than the new path MTU. If IP-level fragmentation can not be used, the SCTP association will be terminated in a non-graceful way. See [RFC8900] for more information about IP fragmentation.

Therefore, a NAT function MUST be able to handle IP-level fragmented SCTP packets. The fragments MAY arrive in any order.

When an SCTP packet can not be forwarded by the NAT function due to MTU issues and the IP header forbids fragmentation, the NAT MUST send back a "Fragmentation needed and DF set" ICMPv4 or PTB ICMPv6 message to the internal host. This allows for a faster recovery from this packet drop.

6.6. Multi Point Traversal Considerations for Endpoints

If a multi-homed SCTP endpoint behind a NAT function connects to a peer, it MUST first set up the association single-homed with only one address causing the first NAT function to populate its state. Then it SHOULD add each IP address using packets containing ASCONF chunks sent via their respective NAT functions. The address used in the Add IP address parameter is the wildcard address (0.0.0.0 or ::0) and the address parameter in the ASCONF chunk SHOULD also contain the VTags parameter and optionally the Disable Restart parameter.

7. SCTP NAT YANG Module

This section defines a YANG module for SCTP NAT.

The terminology for describing YANG data models is defined in [RFC7950]. The meaning of the symbols in tree diagrams is defined in [RFC8340].

7.1. Tree Structure

This module augments NAT YANG module [RFC8512] with SCTP specifics. The module supports both classical SCTP NAT (that is, rewrite port numbers) and SCTP-specific variant where the ports numbers are not altered. The YANG "feature" is used to indicate whether SCTP-specific variant is supported.

The tree structure of the SCTP NAT YANG module is provided below:

```
module: ietf-nat-sctp
  augment /nat:nat/nat:instances/nat:instance
          /nat:policy/nat:timers:
    +--rw sctp-timeout?   uint32
  augment /nat:nat/nat:instances/nat:instance
         /nat:mapping-table/nat:mapping-entry:
    +--rw int-VTag?   uint32 {sctp-nat}?
    +--rw rem-VTag?   uint32 {sctp-nat}?
```

Concretely, the SCTP NAT YANG module augments the NAT YANG module (policy, in particular) with the following:

* The sctp-timeout is used to control the SCTP inactivity timeout. That is, the time an SCTP mapping will stay active without SCTP packets traversing the NAT. This timeout can be set only for SCTP. Hence, "/nat:nat/nat:instances/nat:instance/nat:policy/nat:transport-protocols/nat:protocol-id" MUST be set to '132' (SCTP).

In addition, the SCTP NAT YANG module augments the mapping entry with the following parameters defined in Section 3. These parameters apply only for SCTP NAT mapping entries (i.e., "/nat/instances/instance/mapping-table/mapping-entry/transport-protocol" MUST be set to '132');

* The Internal Verification Tag (Int-VTag)

* The Remote Verification Tag (Rem-VTag)

7.2. YANG Module

```
<CODE BEGINS> file "ietf-nat-sctp@2020-11-02.yang"
module ietf-nat-sctp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nat-sctp";
  prefix nat-sctp;

  import ietf-nat {
    prefix nat;
    reference
      "RFC 8512: A YANG Module for Network Address Translation
                 (NAT) and Network Prefix Translation (NPT)";
  }

  organization
    "IETF TSVWG Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/tsvwg/>
```

```
        WG List:  <mailto:tsvwg@ietf.org>

        Author:  Mohamed Boucadair
                 <mailto:mohamed.boucadair@orange.com>";
     description
       "This module augments NAT YANG module with Stream Control
        Transmission Protocol (SCTP) specifics. The extension supports
        both a classical SCTP NAT (that is, rewrite port numbers)
        and a, SCTP-specific variant where the ports numbers are
        not altered.

        Copyright (c) 2020 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     revision 2019-11-18 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: Stream Control Transmission Protocol (SCTP)
                    Network Address Translation Support";
     }

     feature sctp-nat {
       description
         "This feature means that SCTP-specific variant of NAT
          is supported. That is, avoid rewriting port numbers.";
       reference
         "Section 4.3 of RFC XXXX.";
     }

     augment "/nat:nat/nat:instances/nat:instance"
           + "/nat:policy/nat:timers" {
       when "/nat:nat/nat:instances/nat:instance"
          + "/nat:policy/nat:transport-protocols"
          + "/nat:protocol-id = 132";
       description
         "Extends NAT policy with a timeout for SCTP mapping
          entries.";
```

```
      leaf sctp-timeout {
        type uint32;
        units "seconds";
        description
          "SCTP inactivity timeout. That is, the time an SCTP
           mapping entry will stay active without packets
           traversing the NAT.";
      }
    }

    augment "/nat:nat/nat:instances/nat:instance"
          + "/nat:mapping-table/nat:mapping-entry" {
      when "nat:transport-protocol = 132";
      if-feature "sctp-nat";
      description
        "Extends the mapping entry with SCTP specifics.";

      leaf int-VTag {
         type uint32;
         description
           "The Internal Verification Tag that the internal
            host has chosen for this communication.";
      }
      leaf rem-VTag {
        type uint32;
        description
           "The Remote Verification Tag that the remote
            peer has chosen for this communication.";
      }
    }
  }
  <CODE ENDS>
```

8. Various Examples of NAT Traversals

Please note that this section is informational only.

The addresses being used in the following examples are IPv4 addresses for private-use networks and for documentation as specified in [RFC6890]. However, the method described here is not limited to this NAT44 case.

The NAT binding table entries shown in the following examples do not include the flag indicating whether the restart procedure is supported or not. This flag is not relevant for these examples.

8.1. Single-homed Client to Single-homed Server

The internal client starts the association with the remote server via a four-way-handshake. Host A starts by sending a packet containing an INIT chunk.

```
                                          /--\/--\
+--------+          +-----+            /        \           +--------+
| Host A | <------> | NAT | <------> | Network  | <------> | Host B |
+--------+          +-----+            \        /           +--------+
                                          \--/\---/
       +---------+--------+----------+--------+-----------+
NAT    |  Int    |  Int   |   Rem    |   Rem  |    Int    |
       |  VTag   |  Port  |   VTag   |   Port |    Addr   |
       +---------+--------+----------+--------+-----------+

   INIT[Initiate-Tag = 1234]
10.0.0.1:1 ------> 203.0.113.1:2
        Rem-VTtag = 0
```

A NAT binding tabled entry is created, the source address is substituted and the packet is sent on:

```
       NAT function creates entry:
       +---------+--------+----------+--------+-----------+
NAT    |  Int    |  Int   |   Rem    |   Rem  |    Int    |
       |  VTag   |  Port  |   VTag   |   Port |    Addr   |
       +---------+--------+----------+--------+-----------+
       |  1234   |   1    |    0     |   2    |  10.0.0.1 |
       +---------+--------+----------+--------+-----------+

                                INIT[Initiate-Tag = 1234]
                   192.0.2.1:1 ------------------------> 203.0.113.1:2
                                         Rem-VTtag = 0
```

Host B receives the packet containing an INIT chunk and sends a packet containing an INIT ACK chunk with the NAT's Remote-address as destination address.

```
                                          /--\/--\
   +--------+          +-----+           /        \           +--------+
   | Host A | <------> | NAT | <------> | Network  | <------> | Host B |
   +--------+          +-----+           \        /           +--------+
                                          \--/\---/

                                     INIT ACK[Initiate-Tag = 5678]
                       192.0.2.1:1 <------------------------ 203.0.113.1:2
                                                Int-VTag = 1234
```

NAT function updates entry:

NAT	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	5678	2	10.0.0.1

```
   INIT ACK[Initiate-Tag = 5678]
   10.0.0.1:1 <------ 203.0.113.1:2
             Int-VTag = 1234
```

The handshake finishes with a COOKIE ECHO acknowledged by a COOKIE ACK.

```
                                          /--\/--\
   +--------+          +-----+           /        \           +--------+
   | Host A | <------> | NAT | <------> | Network  | <------> | Host B |
   +--------+          +-----+           \        /           +--------+
                                          \--/\---/

            COOKIE ECHO
   10.0.0.1:1 ------> 203.0.113.1:2
          Rem-VTag = 5678

                                       COOKIE ECHO
                       192.0.2.1:1 ------------------------> 203.0.113.1:2
                                     Rem-VTag = 5678


                                        COOKIE ACK
                       192.0.2.1:1 <------------------------ 203.0.113.1:2
                                     Int-VTag = 1234

               COOKIE ACK
   10.0.0.1:1 <------ 203.0.113.1:2
              Int-VTag = 1234
```

8.2. Single-homed Client to Multi-homed Server

The internal client is single-homed whereas the remote server is multi-homed. The client (Host A) sends a packet containing an INIT chunk like in the single-homed case.

```
                                                  +--------+
                                 /--\/--\      /-|Router 1| \
+------+         +-----+      /        \    /  +--------+  \ +------+
| Host | <-----> | NAT | <-> | Network  | ==                =| Host |
|   A  |         +-----+      \        /    \  +--------+  / |   B  |
+------+                       \--/\--/      \-|Router 2|-/  +------+
                                                  +--------+
```

```
       +---------+--------+----------+--------+-----------+
NAT    |  Int    |  Int   |   Rem    |   Rem  |    Int    |
       |  VTag   |  Port  |   VTag   |   Port |    Addr   |
       +---------+--------+----------+--------+-----------+
```

```
 INIT[Initiate-Tag = 1234]
10.0.0.1:1 ---> 203.0.113.1:2
       Rem-VTag = 0
```

NAT function creates entry:

```
       +---------+--------+----------+--------+-----------+
NAT    |  Int    |  Int   |   Rem    |   Rem  |    Int    |
       |  VTag   |  Port  |   VTag   |   Port |    Addr   |
       +---------+--------+----------+--------+-----------+
       |  1234   |   1    |    0     |   2    |  10.0.0.1 |
       +---------+--------+----------+--------+-----------+
```

```
                              INIT[Initiate-Tag = 1234]
                 192.0.2.1:1 -------------------------> 203.0.113.1:2
                                   Rem-VTag = 0
```

The server (Host B) includes its two addresses in the INIT ACK chunk.

```
                                                  +--------+
                                  /--\/--\      /-|Router 1| \
   +------+         +-----+      /        \    /  +--------+  \ +------+
   | Host | <-----> | NAT | <-> | Network  | ==                =| Host |
   |   A  |         +-----+      \        /    \  +--------+  / |   B  |
   +------+                       \--/\--/      \-|Router 2|-/  +------+
                                                  +--------+

                  INIT ACK[Initiate-tag = 5678, IP-Addr = 203.0.113.129]
                   192.0.2.1:1 <-------------------------- 203.0.113.1:2
                                     Int-VTag = 1234
```

The NAT function does not need to change the NAT binding table for the second address:

NAT	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	5678	2	10.0.0.1

```
     INIT ACK[Initiate-Tag = 5678]
     10.0.0.1:1 <--- 203.0.113.1:2
              Int-VTag = 1234
```

The handshake finishes with a COOKIE ECHO acknowledged by a COOKIE ACK.

```
                                                  +--------+
                                     /--\/--\    /-|Router 1| \
   +------+         +-----+        /        \   /  +--------+  \ +------+
   | Host | <-----> | NAT | <-> | Network  | ==                =| Host |
   |   A  |         +-----+        \        /   \  +--------+  / |   B  |
   +------+                         \--/\--/     \-|Router 2|-/  +------+
                                                  +--------+

          COOKIE ECHO
   10.0.0.1:1 ---> 203.0.113.1:2
          Rem-VTag = 5678

                                     COOKIE ECHO
                   192.0.2.1:1 --------------------------> 203.0.113.1:2
                                    Rem-VTag = 5678


                                      COOKIE ACK
                   192.0.2.1:1 <------------------------- 203.0.113.1:2
                                     Int-VTag = 1234

             COOKIE ACK
   10.0.0.1:1 <--- 203.0.113.1:2
            Int-VTag = 1234
```

8.3. Multihomed Client and Server

The client (Host A) sends a packet containing an INIT chunk to the server (Host B), but does not include the second address.

```
                      +-------+
                   /--| NAT 1 |--\       /--\/--\
        +------+   /   +-------+   \     /        \     +--------+
        | Host |===                 ====| Network  |====| Host B |
        |   A  |   \   +-------+   /     \        /     +--------+
        +------+    \--| NAT 2 |--/       \--/\--/
                      +-------+

               +---------+--------+----------+--------+-----------+
        NAT 1  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
               |  VTag   |  Port  |   VTag   |   Port |    Addr   |
               +---------+--------+----------+--------+-----------+

         INIT[Initiate-Tag = 1234]
        10.0.0.1:1 --------> 203.0.113.1:2
                 Rem-VTag = 0
```

NAT function 1 creates entry:

```
        +---------+--------+----------+--------+-----------+
NAT 1   |  Int    |  Int   |   Rem    |   Rem  |    Int    |
        |  VTag   |  Port  |   VTag   |   Port |    Addr   |
        +---------+--------+----------+--------+-----------+
        |  1234   |    1   |     0    |    2   |  10.0.0.1 |
        +---------+--------+----------+--------+-----------+


                                INIT[Initiate-Tag = 1234]
                      192.0.2.1:1 ---------------------> 203.0.113.1:2
                                          Rem-VTag = 0
```

Host B includes its second address in the INIT ACK.

```
                      +-------+
            /--------| NAT 1 |--------\       /--\/--\
+------+   /         +-------+         \     /        \    +--------+
| Host |===                              ====| Network  |===| Host B |
|   A  |   \         +-------+         /     \        /    +--------+
+------+    \--------| NAT 2 |--------/       \--/\--/
                      +-------+

                INIT ACK[Initiate-Tag = 5678, IP-Addr = 203.0.113.129]
                    192.0.2.1:1 <----------------------- 203.0.113.1:2
                                     Int-VTag = 1234
```

NAT function 1 does not need to update the NAT binding table for the second address:

```
            +---------+--------+----------+--------+-----------+
     NAT 1  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
            |  VTag   |  Port  |   VTag   |   Port |    Addr   |
            +---------+--------+----------+--------+-----------+
            |  1234   |    1   |    5678  |    2   |  10.0.0.1 |
            +---------+--------+----------+--------+-----------+


       INIT ACK[Initiate-Tag = 5678]
     10.0.0.1:1 <-------- 203.0.113.1:2
                  Int-VTag = 1234
```

The handshake finishes with a COOKIE ECHO acknowledged by a COOKIE ACK.

```
                        +-------+
              /--------| NAT 1 |--------\       /--\/--\
   +------+   /         +-------+         \     /        \    +--------+
   | Host |===                              ====| Network  |===| Host B |
   |   A  |   \         +-------+         /     \        /    +--------+
   +------+    \--------| NAT 2 |--------/       \--/\--/
                        +-------+

             COOKIE ECHO
   10.0.0.1:1 --------> 203.0.113.1:2
             Rem-VTag = 5678

                                              COOKIE ECHO
                              192.0.2.1:1 ------------------> 203.0.113.1:2
                                            Rem-VTag = 5678


                                              COOKIE ACK
                              192.0.2.1:1 <------------------ 203.0.113.1:2
                                            Int-VTag = 1234

              COOKIE ACK
   10.0.0.1:1 <------- 203.0.113.1:2
             Int-VTag = 1234
```

Host A announces its second address in an ASCONF chunk. The address parameter contains a wildcard address (0.0.0.0 or ::0) to indicate that the source address has to be be added. The address parameter within the ASCONF chunk will also contain the pair of VTags (remote and internal) so that the NAT function can populate its NAT binding table entry completely with this single packet.

```
                        +-------+
              /--------| NAT 1 |--------\       /--\/--\
   +------+   /         +-------+         \     /        \    +--------+
   | Host |===                              ====| Network  |===| Host B |
   |   A  |   \         +-------+         /     \        /    +--------+
   +------+    \--------| NAT 2 |--------/       \--/\--/
                        +-------+

   ASCONF [ADD-IP=0.0.0.0, INT-VTag=1234, Rem-VTag = 5678]
   10.1.0.1:1 --------> 203.0.113.129:2
            Rem-VTag = 5678
```

NAT function 2 creates a complete entry:

```
          +---------+--------+----------+--------+-----------+
   NAT 2  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
          |  VTag   |  Port  |   VTag   |   Port |    Addr   |
          +---------+--------+----------+--------+-----------+
          |  1234   |    1   |    5678  |    2   |  10.1.0.1 |
          +---------+--------+----------+--------+-----------+


                     ASCONF [ADD-IP, Int-VTag=1234, Rem-VTag = 5678]
                      192.0.2.129:1 -------------------> 203.0.113.129:2
                                         Rem-VTag = 5678

                                            ASCONF ACK
                      192.0.2.129:1 <------------------- 203.0.113.129:2
                                         Int-VTag = 1234

             ASCONF ACK
   10.1.0.1:1 <----- 203.0.113.129:2
            Int-VTag = 1234
```

8.4. NAT Function Loses Its State

Association is already established between Host A and Host B, when the NAT function loses its state and obtains a new external address. Host A sends a DATA chunk to Host B.

```
                                            /--\/--\
   +--------+              +-----+         /        \         +--------+
   | Host A | <----------> | NAT | <----> | Network  | <----> | Host B |
   +--------+              +-----+         \        /         +--------+
                                            \--/\--/

          +---------+--------+----------+--------+-----------+
   NAT    |  Int    |  Int   |   Rem    |   Rem  |    Int    |
          |  VTag   |  Port  |   VTag   |   Port |    Addr   |
          +---------+--------+----------+--------+-----------+

                  DATA
   10.0.0.1:1 ----------> 203.0.113.1:2
               Rem-VTag = 5678
```

The NAT function cannot find an entry in the NAT binding table for the association. It sends a packet containing an ERROR chunk with the M bit set and the cause "NAT state missing".

```
                                          /--\/--\
+--------+              +-----+         /        \          +--------+
| Host A | <----------> | NAT | <----> | Network  | <----> | Host B |
+--------+              +-----+         \        /          +--------+
                                          \--/\--/

  ERROR [M bit, NAT state missing]
10.0.0.1:1 <---------- 203.0.113.1:2
          Rem-VTag = 5678
```

On reception of the packet containing the ERROR chunk, Host A sends a packet containing an ASCONF chunk indicating that the former information has to be deleted and the source address of the actual packet added.

```
                                          /--\/--\
+--------+              +-----+         /        \          +--------+
| Host A | <----------> | NAT | <----> | Network  | <----> | Host B |
+--------+              +-----+         \        /          +--------+
                                          \--/\--/

ASCONF [ADD-IP, DELETE-IP, Int-VTag=1234, Rem-VTag = 5678]
10.0.0.1:1 ----------> 203.0.113.129:2
          Rem-VTag = 5678

       +---------+--------+----------+--------+-----------+
NAT    |  Int    |  Int   |   Rem    |   Rem  |    Int    |
       |  VTag   |  Port  |   VTag   |   Port |    Addr   |
       +---------+--------+----------+--------+-----------+
       |  1234   |    1   |    5678  |    2   |  10.0.0.1 |
       +---------+--------+----------+--------+-----------+

        ASCONF [ADD-IP, DELETE-IP, Int-VTag=1234, Rem-VTag = 5678]
                       192.0.2.2:1 -----------------> 203.0.113.129:2
                                     Rem-VTag = 5678
```

Host B adds the new source address to this association and deletes all other addresses from this association.

```
                                          /--\/--\
   +--------+              +-----+       /        \         +--------+
   | Host A | <----------> | NAT | <----> | Network  | <----> | Host B |
   +--------+              +-----+       \        /         +--------+
                                          \--/\--/

                                          ASCONF ACK
                         192.0.2.2:1 <----------------- 203.0.113.129:2
                                       Int-VTag = 1234

             ASCONF ACK
   10.1.0.1:1 <---------- 203.0.113.129:2
             Int-VTag = 1234

               DATA
   10.0.0.1:1 ----------> 203.0.113.1:2
            Rem-VTag = 5678
                                           DATA
                         192.0.2.2:1 ----------------> 203.0.113.129:2
                                      Rem-VTag = 5678
```

8.5. Peer-to-Peer Communications

If two hosts, each of them behind a NAT function, want to communicate with each other, they have to get knowledge of the peer's external address. This can be achieved with a so-called rendezvous server. Afterwards the destination addresses are external, and the association is set up with the help of the INIT collision. The NAT functions create their entries according to their internal peer's point of view. Therefore, NAT function A's Internal-VTag and Internal-Port are NAT function B's Remote-VTag and Remote-Port, respectively. The naming (internal/remote) of the verification tag in the packet flow is done from the sending host's point of view.

```
             Internal | External           External | Internal
                      |                            |
                      |          /--\/---\         |
   +--------+     +-------+     /         \     +-------+     +--------+
   | Host A |<--->| NAT A |<-->| Network   |<-->| NAT B |<--->| Host B |
   +--------+     +-------+     \         /     +-------+     +--------+
                      |          \--/\---/         |
```

NAT Binding Tables

```
          +---------+--------+----------+--------+-----------+
   NAT A  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
          |  VTag   |  Port  |   VTag   |   Port |    Addr   |
          +---------+--------+----------+--------+-----------+

          +---------+--------+----------+--------+-----------+
   NAT B  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
          |  v-tag  |  port  |   v-tag  |   port |    Addr   |
          +---------+--------+----------+--------+-----------+
```

```
   INIT[Initiate-Tag = 1234]
   10.0.0.1:1 --> 203.0.113.1:2
           Rem-VTag = 0
```

NAT function A creates entry:

```
                +---------+--------+----------+--------+-----------+
        NAT A   |  Int    |  Int   |   Rem    |   Rem  |    Int    |
                |  VTag   |  Port  |   VTag   |   Port |    Addr   |
                +---------+--------+----------+--------+-----------+
                |  1234   |   1    |    0     |   2    |  10.0.0.1 |
                +---------+--------+----------+--------+-----------+
```

```
                          INIT[Initiate-Tag = 1234]
                192.0.2.1:1 ----------------> 203.0.113.1:2
                             Rem-VTag = 0
```

NAT function B processes the packet containing the INIT chunk, but cannot find an entry. The SCTP packet is silently discarded and leaves the NAT binding table of NAT function B unchanged.

```
                +---------+--------+----------+--------+-----------+
        NAT B   |  Int    |  Int   |   Rem    |   Rem  |    Int    |
                |  VTag   |  Port  |   VTag   |   Port |    Addr   |
                +---------+--------+----------+--------+-----------+
```

Now Host B sends a packet containing an INIT chunk, which is processed by NAT function B. Its parameters are used to create an entry.

```
          Internal | External           External | Internal
                   |                            |
                   |          /--\/---\         |
+--------+     +-------+     /         \     +-------+     +--------+
| Host A |<--->| NAT A |<-->| Network   |<-->| NAT B |<--->| Host B |
+--------+     +-------+     \         /     +-------+     +--------+
                   |          \--/\---/         |

                                          INIT[Initiate-Tag = 5678]
                                         192.0.2.1:1 <-- 10.1.0.1:2
                                                       Rem-VTag = 0
```

NAT B

Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
5678	2	0	1	10.1.0.1

```
                     INIT[Initiate-Tag = 5678]
          192.0.2.1:1  <--------------- 203.0.113.1:2
                               Rem-VTag = 0
```

NAT function A processes the packet containing the INIT chunk. As the outgoing packet containing an INIT chunk of Host A has already created an entry, the entry is found and updated:

```
             Internal | External           External | Internal
                      |                            |
                      |          /--\/---\         |
   +--------+     +-------+     /         \     +-------+     +--------+
   | Host A |<--->| NAT A |<-->| Network   |<-->| NAT B |<--->| Host B |
   +--------+     +-------+     \         /     +-------+     +--------+
                      |          \--/\---/         |
```

VTag != Int-VTag, but Rem-VTag == 0, find entry.

	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
NAT A	1234	1	5678	2	10.0.0.1

```
    INIT[Initiate-tag = 5678]
   10.0.0.1:1 <-- 203.0.113.1:2
             Rem-VTag = 0
```

Host A sends a packet containing an INIT ACK chunk, which can pass through NAT function B:

```
             Internal | External           External | Internal
                      |                            |
                      |          /--\/---\         |
   +--------+     +-------+     /         \     +-------+     +--------+
   | Host A |<--->| NAT A |<-->| Network   |<-->| NAT B |<--->| Host B |
   +--------+     +-------+     \         /     +-------+     +--------+
                      |          \--/\---/         |

   INIT ACK[Initiate-Tag = 1234]
   10.0.0.1:1 --> 203.0.113.1:2
         Rem-VTag = 5678


                        INIT ACK[Initiate-Tag = 1234]
                  192.0.2.1:1 ----------------> 203.0.113.1:2
                                Rem-VTag = 5678

                                       NAT function B updates entry:

          +---------+--------+----------+--------+-----------+
   NAT B  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
          |  VTag   |  Port  |   VTag   |   Port |    Addr   |
          +---------+--------+----------+--------+-----------+
          |  5678   |    2   |   1234   |   1    |  10.1.0.1 |
          +---------+--------+----------+--------+-----------+

                                          INIT ACK[Initiate-Tag = 1234]
                                             192.0.2.1:1 --> 10.1.0.1:2
                                                        Rem-VTag = 5678
```

The lookup for COOKIE ECHO and COOKIE ACK is successful.

```
             Internal | External           External | Internal
                      |                            |
                      |          /--\/---\         |
   +--------+     +-------+     /         \     +-------+     +--------+
   | Host A |<--->| NAT A |<-->| Network   |<-->| NAT B |<--->| Host B |
   +--------+     +-------+     \         /     +-------+     +--------+
                      |          \--/\---/         |

                                                     COOKIE ECHO
                                            192.0.2.1:1 <-- 10.1.0.1:2
                                                 Rem-VTag = 1234

                                 COOKIE ECHO
                 192.0.2.1:1 <------------- 203.0.113.1:2
                                 Rem-VTag = 1234

          COOKIE ECHO
   10.0.0.1:1 <-- 203.0.113.1:2
          Rem-VTag = 1234

          COOKIE ACK
   10.0.0.1:1 --> 203.0.113.1:2
          Rem-VTag = 5678

                                 COOKIE ACK
                 192.0.2.1:1 ----------------> 203.0.113.1:2
                                 Rem-VTag = 5678

                                                       COOKIE ACK
                                            192.0.2.1:1 --> 10.1.0.1:2
                                                  Rem-VTag = 5678
```

9. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to control NAT friendliness.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is extended by supporting one new read/write socket option.

9.1. Get or Set the NAT Friendliness (SCTP_NAT_FRIENDLY)

This socket option uses the option_level IPPROTO_SCTP and the option_name SCTP_NAT_FRIENDLY. It can be used to enable/disable the NAT friendliness for future associations and retrieve the value for future and specific ones.

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id
: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application can fill in an association identifier or SCTP_FUTURE_ASSOC for this query. It is an error to use SCTP_{CURRENT|ALL}_ASSOC in assoc_id.

assoc_value
: A non-zero value indicates a NAT-friendly mode.

10. IANA Considerations

[NOTE to RFC-Editor: "RFCXXXX" is to be replaced by the RFC number you assign this document.]

[NOTE to RFC-Editor: The requested values for the chunk type and the chunk parameter types are tentative and to be confirmed by IANA.]

This document (RFCXXXX) is the reference for all registrations described in this section. The requested changes are described below.

10.1. New Chunk Flags for Two Existing Chunk Types

As defined in [RFC6096] two chunk flags have to be assigned by IANA for the ERROR chunk. The requested value for the T bit is 0x01 and for the M bit is 0x02.

This requires an update of the "ERROR Chunk Flags" registry for SCTP:

ERROR Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	T bit	[RFCXXXX]
0x02	M bit	[RFCXXXX]
0x04	Unassigned	
0x08	Unassigned	
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

Table 2

As defined in [RFC6096] one chunk flag has to be assigned by IANA for the ABORT chunk. The requested value of the M bit is 0x02.

This requires an update of the "ABORT Chunk Flags" registry for SCTP:

ABORT Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	T bit	[RFC4960]
0x02	M bit	[RFCXXXX]
0x04	Unassigned	
0x08	Unassigned	
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

Table 3

10.2. Three New Error Causes

Three error causes have to be assigned by IANA. It is requested to use the values given below.

This requires three additional lines in the "Error Cause Codes" registry for SCTP:

Error Cause Codes

Value	Cause Code	Reference
176	VTag and Port Number Collision	[RFCXXXX]
177	Missing State	[RFCXXXX]
178	Port Number Collision	[RFCXXXX]

Table 4

10.3. Two New Chunk Parameter Types

Two chunk parameter types have to be assigned by IANA. IANA is requested to assign these values from the pool of parameters with the upper two bits set to '11' and to use the values given below.

This requires two additional lines in the "Chunk Parameter Types" registry for SCTP:

Chunk Parameter Types

ID Value	Chunk Parameter Type	Reference
49159	Disable Restart (0xC007)	[RFCXXXX]
49160	VTags (0xC008)	[RFCXXXX]

Table 5

10.4. One New URI

An URI in the "ns" subregistry within the "IETF XML" registry has to be assigned by IANA ([RFC3688]):

URI: urn:ietf:params:xml:ns:yang:ietf-nat-sctp
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

10.5. One New YANG Module

An YANG module in the "YANG Module Names" subregistry within the "YANG Parameters" registry has to be assigned by IANA ([RFC6020]):

Name: ietf-nat-sctp
Namespace: urn:ietf:params:xml:ns:yang:ietf-nat-sctp
Maintained by IANA: N
Prefix: nat-sctp
Reference: RFCXXXX

11. Security Considerations

State maintenance within a NAT function is always a subject of possible Denial Of Service attacks. This document recommends that at a minimum a NAT function runs a timer on any SCTP state so that old association state can be cleaned up.

Generic issues related to address sharing are discussed in [RFC6269] and apply to SCTP as well.

For SCTP endpoints not disabling the restart procedure, this document does not add any additional security considerations to the ones given in [RFC4960], [RFC4895], and [RFC5061].

SCTP endpoints disabling the restart procedure, need to monitor the status of all associations to mitigate resource exhaustion attacks by establishing a lot of associations sharing the same IP addresses and port numbers.

In any case, SCTP is protected by the verification tags and the usage of [RFC4895] against off-path attackers.

For IP-level fragmentation and reassembly related issues see [RFC4963].

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All data nodes defined in the YANG module that can be created, modified, and deleted (i.e., config true, which is the default) are considered sensitive. Write operations (e.g., edit-config) applied to these data nodes without proper protection can negatively affect network operations. An attacker who is able to access the SCTP NAT function can undertake various attacks, such as:

* Setting a low timeout for SCTP mapping entries to cause failures to deliver incoming SCTP packets.

* Instantiating mapping entries to cause NAT collision.

12. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, <https://www.rfc-editor.org/info/rfc4895>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <https://www.rfc-editor.org/info/rfc5061>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, DOI 10.17487/RFC6096, January 2011, <https://www.rfc-editor.org/info/rfc6096>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <https://www.rfc-editor.org/info/rfc8512>.

13. Informative References

[DOI_10.1145_1496091.1496095]
Hayes, D., But, J., and G. Armitage, "Issues with network address translation for SCTP", ACM SIGCOMM Computer Communication Review Vol. 39, pp. 23-33, DOI 10.1145/1496091.1496095, December 2008, <https://doi.org/10.1145/1496091.1496095>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <https://www.rfc-editor.org/info/rfc3022>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <https://www.rfc-editor.org/info/rfc4787>.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <https://www.rfc-editor.org/info/rfc4963>.

[RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008, <https://www.rfc-editor.org/info/rfc5382>.

[RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, DOI 10.17487/RFC5508, April 2009, <https://www.rfc-editor.org/info/rfc5508>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <https://www.rfc-editor.org/info/rfc6056>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <https://www.rfc-editor.org/info/rfc6269>.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <https://www.rfc-editor.org/info/rfc6333>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <https://www.rfc-editor.org/info/rfc6458>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <https://www.rfc-editor.org/info/rfc6951>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar, S., and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements", BCP 127, RFC 7857, DOI 10.17487/RFC7857, April 2016, <https://www.rfc-editor.org/info/rfc7857>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <https://www.rfc-editor.org/info/rfc8900>.

Acknowledgments

The authors wish to thank Mohamed Boucadair, Gorry Fairhurst, Bryan Ford, David Hayes, Alfred Hines, Karen E. E. Nielsen, Henning Peters, Maksim Proshin, Timo Völker, Dan Wing, and Qiaobing Xie for their invaluable comments.

In addition, the authors wish to thank David Hayes, Jason But, and Grenville Armitage, the authors of [DOI_10.1145_1496091.1496095], for their suggestions.

The authors also wish to thank Mohamed Boucadair for contributing the text related to the YANG module.

Authors' Addresses

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
United States of America

Email: randall@lakerest.net

Michael Tüxen
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Irene Rüngeler
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: i.ruengeler@fh-muenster.de

TSVWG
Internet Draft
Intended status: Best Current Practice
Expires: October 2015

J. Touch
USC/ISI

April 24, 2015

Recommendations on Using Assigned Transport Port Numbers
draft-ietf-tsvwg-port-use-11.txt

Status of this Memo

Copyright Notice

Abstract

This document provides recommendations to application and service protocol designers on how to use the assigned transport protocol port number space and when to request a port assignment from IANA. It provides designer guidelines on how to interact with the IANA processes defined in RFC6335, thus serving to complement (but not update) that document.

Table of Contents

1. Introduction

This document provides information and advice to application and service designers on the use of assigned transport port numbers. It provides a detailed historical background of the evolution of transport port numbers and their multiple meanings. It also provides specific recommendations to designers on how to use assigned port numbers. Note that this document provides information to potential port number applicants that complements the IANA process described in BCP165 [RFC6335], but it does not change any of the port number

assignment procedures described therein. This document is intended to address concerns typically raised during Expert Review of assigned port number applications, but it is not intended to bind those reviews. RFC 6335 also describes the interaction between port experts and port requests in IETF consensus document. Authors of IETF consensus documents should nevertheless follow the advice in this document and can expect comment on their port requests from the port experts during IETF last call or at other times when review is explicitly sought.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s) indicates a statement using the key words listed above. This convention aids reviewers in quickly identifying or finding requirements for registration and recommendations for use of port numbers in this RFC.

3. History

The term ’port’ was first used in [RFC33] to indicate a simplex communication path from an individual process and originally applied to only the Network Control Program (NCP) connection-oriented protocol. At a meeting described in [RFC37], an idea was presented to decouple connections between processes and links that they use as paths, and thus to include numeric source and destination socket identifiers in packets. [RFC38] provides further detail, describing how processes might have more than one of these paths and that more than one path may be active at a time. As a result, there was the need to add a process identifier to the header of each message so that incoming messages could be demultiplexed to the appropriate process. [RFC38] further suggested that 32 bit numbers would be used for these identifiers. [RFC48] discusses the current notion of listening on a specific port number, but does not discuss the issue of port number determination. [RFC61] notes that the challenge of knowing the appropriate port numbers is "left to the processes" in general, but introduces the concept of a "well-known" port number for common services.

[RFC76] proposed a "telephone book" by which an index would allow port numbers to be used by name, but still assumed that both source and destination port numbers are fixed by such a system. [RFC333] proposed that a port number pair, rather than an individual port number, would be used on both sides of the connection for demultiplexing messages. This is the final view in [RFC793] (and its predecessors, including [IEN112]), and brings us to their current meaning. [RFC739] introduced the notion of generic reserved port numbers for groups of protocols, such as "any private RJE server" [RFC739]. Although the overall range of such port numbers was (and remains) 16 bits, only the first 256 (high 8 bits cleared) in the range were considered assigned.

[RFC758] is the first to describe port numbers as being used for TCP (previous RFCs all refer to only NCP). It includes a list of such well-known port numbers, as well as describing ranges used for different purposes:

Decimal	Octal	
0-63	0-77	Network Wide Standard Function
64-127	100-177	Hosts Specific Functions
128-223	200-337	Reserved for Future Use
224-255	340-377	Any Experimental Function

In [RFC820] those range meanings disappeared, and a single list of number assignments is presented. This is also the first time that port numbers are described as applying to a connectionless transport (UDP) rather than only connection-oriented transports.

By [RFC900] the ranges appeared as decimal numbers rather than the octal ranges used previously. [RFC1340] increased this range from 0..255 to 0..1023, and began to list TCP and UDP port number assignments individually (although the assumption was that once assigned a port number applies to all transport protocols, including TCP, UDP, recently SCTP and DCCP, as well as ISO-TP4 for a brief period in the early 1990s). [RFC1340] also established the Registered range of 1024-59151, though it notes that it is not controlled by the IANA at that point. The list provided by [RFC1700] in 1994 remained the standard until it was declared replaced by an on-line version, as of [RFC3232] in 2002.

4. Current Port Number Use

RFC6335 indicates three ranges of port number assignments:

Binary	Hex	
0-1023	0x0000-0x03FF	System (also Well-Known)
1024-49151	0x0400-0xBFFF	User (also Registered)
49152-65535	0xC000-0xFFFF	Dynamic (also Private)

System (also Well-Known) encompasses the range 0..1023. On some systems, use of these port numbers requires privileged access, e.g., that the process run as ’root’ (i.e., as a privileged user), which is why these are referred to as System port numbers. The port numbers from 1024..49151 denotes non-privileged services, known as User (also Registered), because these port numbers do not run with special privileges. Dynamic (also Private) port numbers are not assigned.

Both System and User port numbers are assigned through IANA, so both are sometimes called ’registered port numbers’. As a result, the term ’registered’ is ambiguous, referring either to the entire range 0-49151 or to the User port numbers. Complicating matters further, System port numbers do not always require special (i.e., ’root’) privilege. For clarity, the remainder of this document refers to the port number ranges as System, User, and Dynamic, to be consistent with IANA process [RFC6335].

5. What is a Port Number?

A port number is a 16-bit number used for two distinct purposes:

- o Demultiplexing transport endpoint associations within an end host
- o Identifying a service

The first purpose requires that each transport endpoint association (e.g., TCP connection or UDP pairwise association) using a given transport between a given pair of IP addresses use a different pair of port numbers, but does not require either coordination or registration of port number use. It is the second purpose that drives the need for a common registry.

Consider a user wanting to run a web server. That service could run on any port number, provided that all clients knew what port number to use to access that service at that host. Such information can be explicitly distributed - for example, by putting it in the URI:

http://www.example.com:51509/

Ultimately, the correlation of a service with a port number is an agreement between just the two endpoints of the association. A web server can run on port number 53, which might appear as DNS traffic to others but will connect to browsers that know to use port number 53 rather than 80.

As a concept, a service is the combination of ISO Layers 5-7 that represents an application protocol capability. For example www (port number 80) is a service that uses HTTP as an application protocol and provides access to a web server [RFC7230]. However, it is possible to use HTTP for other purposes, such as command and control. This is why some current services (HTTP, e.g.) are a bit overloaded - they describe not only the application protocol, but a particular service.

IANA assigns port numbers so that Internet endpoints do not need pairwise, explicit coordination of the meaning of their port numbers. This is the primary reason for requesting port number assignment by IANA - to have a common agreement between all endpoints on the Internet as to the default meaning of a port number, which provides the endpoints with a default port number for a particular protocol or service.

Port numbers are sometimes used by intermediate devices on a network path, either to monitor available services, to monitor traffic (e.g., to indicate the data contents), or to intercept traffic (to block, proxy, relay, aggregate, or otherwise process it). In each case, the intermediate device interprets traffic based on the port number. It is important to recognize that any interpretation of port numbers - except at the endpoints - may be incorrect, because port numbers are meaningful only at the endpoints. Further, port numbers may not be visible to these intermediate devices, such as when the transport protocol is encrypted (as in network- or link-layer tunnels), or when a packet is fragmented (in which case only the first fragment has the port number information). Such port number invisibility may interfere with these in-network port number-based capabilities.

Port numbers can also be used for other purposes. Assigned port numbers can simplify end system configuration, so that individual

installations do not need to coordinate their use of arbitrary port numbers. Such assignments may also have the effect of simplifying firewall management, so that a single, fixed firewall configuration can either permit or deny a service that uses the assigned ports.

It is useful to differentiate a port number from a service name. The former is a numeric value that is used directly in transport protocol headers as a demultiplexing and service identifier. The latter is primarily a user convenience, where the default map between the two is considered static and resolved using a cached index. This document focuses on the former because it is the fundamental network resource. Dynamic maps between the two, i.e., using DNS SRV records, are discussed further in Section 7.1.

6. Conservation

Assigned port numbers are a limited resource that is globally shared by the entire Internet community. As of 2014, approximately 5850 TCP and 5570 UDP port numbers have been assigned out of a total range of 49151. As a result of past conservation, current assigned port use is small and the current rate of assignment avoids the need for transition to larger number spaces. This conservation also helps avoid the need for IANA to rely on assigned port number reclamation, which is practically impossible even though procedurally permitted [RFC6335].

IANA aims to assign only one port number per service, including variants [RFC6335], but there are other benefits to using fewer port numbers for a given service. Use of multiple assigned port numbers can make applications more fragile, especially when firewalls block a subset of those port numbers or use ports numbers to route or prioritize traffic differently. As a result:

>> Each assigned port requested MUST be justified by the applicant as an independently useful service.

6.1. Guiding Principles

This document provides recommendations for users that also help conserve assigned port number space. Again, this document does not update BCP165 [RFC6335], which describes the IANA procedures for managing assigned transport port numbers and services. Assigned port number conservation is based on a number of basic principles:

- o A single assigned port number can support different functions over separate endpoint associations, determined using in-band information. An FTP data connection can transfer binary or text files, the latter translating line-terminators, as indicated in-band over the control port number [RFC959].

- o A single assigned port number can indicate the Dynamic port number(s) on which different capabilities are supported, as with passive-mode FTP [RFC959].

- o Several existing services can indicate the Dynamic port number(s) on which other services are supported, such as with mDNS and portmapper [RFC1833] [RFC6762] [RFC6763].

- o Copies of some existing services can be differentiated using in-band information (e.g., URIs in HTTP Host field and TLS Server Name Indication extension) [RFC7230] [RFC6066].

- o Services requiring varying performance properties can already be supported using separate endpoint associations (connections or other associations), each configured to support the desired properties. E.g., a high-speed and low-speed variant can be determined within the service using the same assigned port.

Assigned port numbers are intended to differentiate services, not variations of performance, replicas, pairwise endpoint associations, or payload types. Assigned port numbers are also a small space compared to other Internet number spaces; it is never appropriate to consume assigned port numbers to conserve larger spaces such as IP addresses, especially where copies of a service represent different endpoints.

6.2. Firewall and NAT Considerations

Ultimately, port numbers numbers indicate services only to the endpoints, and any intermediate device that assigns meaning to a value can be incorrect. End systems might agree to run web services (HTTP) over port number 53 (typically used for DNS) rather than port number 80, at which point a firewall that blocks port number 80 but permits port number 53 would not have the desired effect. Nonetheless, assigned port numbers are often used to help configure firewalls and other port-based systems for access control.

Using Dynamic port numbers, or explicitly-indicated port numbers indicated in-band over another service (such as with FTP) often complicates firewall and NAT interactions [RFC959]. FTP over firewalls often requires direct support for deep-packet inspection

(to snoop for the Dynamic port number for the NAT to correctly map) or passive-mode FTP (in which both connections are opened from the client side).

7. Considerations for Requesting Port Number Assignments

Port numbers are assigned by IANA by a set of documented procedures [RFC6335]. The following section describes the steps users can take to help assist with responsible use of assigned port numbers, and with preparing an application for a port number assignment.

7.1. Is a port number assignment necessary?

First, it is useful to consider whether a port number assignment is required. In many cases, a new number assignment may not be needed, for example:

- Is this really a new service, or can an existing service suffice?

- Is this an experimental service [RFC3692]? If so, consider using the current experimental ports [RFC2780].

- Is this service independently useful? Some systems are composed from collections of different service capabilities, but not all component functions are useful as independent services. Port numbers are typically shared among the smallest independently-useful set of functions. Different service uses or properties can be supported in separate pairwise endpoint associations after an initial negotiation, e.g., to support software decomposition.

- Can this service use a Dynamic port number that is coordinated out-of-band, e.g.:

  - By explicit configuration of both endpoints.

  - By internal mechanisms within the same host (e.g., a configuration file, indicated within a URI, or using interprocess communication).

  - Using information exchanged on a related service: FTP, SIP, etc. [RFC959] [RFC3261].

  - Using an existing port discovery service: portmapper, mDNS, etc. [RFC1833] [RFC6762] [RFC6763].

There are a few good examples of reasons that more directly suggest that not only is a port number assignment not necessary, but it is directly counter-indicated:

- Assigned port numbers are not intended to differentiate performance variations within the same service, e.g., high-speed vs. ordinary speed. Performance variations can be supported within a single assigned port number in context of separate pairwise endpoint associations.

- Additional assigned port numbers are not intended to replicate an existing service. For example, if a device is configured to use a typical web browser then it the port number used for that service is a copy of the http service that is already assigned to port number 80 and does not warrant a new assignment. However, an automated system that happens to use HTTP framing - but is not primarily accessed by a browser - might be a new service. A good way to tell is "can an unmodified client of the existing service interact with the proposed service"? If so, that service would be a copy of an existing service and would not merit a new assignment.

- Assigned port numbers not intended for intra-machine communication. Such communication can already be supported by internal mechanisms (interprocess communication, shared memory, shared files, etc.). When Internet communication within a host is desired, the server can bind to a Dynamic port that is indicated to the client using these internal mechanisms.

- Separate assigned port numbers are not intended for insecure versions of existing (or new) secure services. A service that already requires security would be made more vulnerable by having the same capability accessible without security.

  Note that the converse is different, i.e., it can be useful to create a new, secure service that replicates an existing insecure service on a new port number assignment. This can be necessary when the existing service is not backward-compatible with security enhancements, such as the use of TLS [RFC5246] or DTLS [RFC6347].

- o Assigned port numbers are not intended for indicating different service versions. Version differentiation should be handled in-band, e.g., using a version number at the beginning of an association (e.g., connection or other transaction). This may not be possible with legacy assignments, but all new services should incorporate support for version indication.

Some services may not need assigned port numbers at all, e.g., SIP allows voice calls to use Dynamic ports [RFC3261]. Some systems can register services in the DNS, using SRV entries. These services can be discovered by a variety of means, including mDNS, or via direct query [RFC6762] [RFC6763]. In such cases, users can more easily request a SRV name, which are assigned first-come, first-served from a much larger namespace.

IANA assigns port numbers, but this assignment is typically used only for servers, i.e., the host that listens for incoming connections or other associations. Clients, i.e., hosts that initiate connections or other associations, typically refer to those assigned port numbers but do not need port number assignments for their endpoint.

Finally, an assigned port number is not a guarantee of exclusive use. Traffic for any service might appear on any port number, due to misconfiguration or deliberate misuse. Application and service designers are encouraged to validate traffic based on its content.

7.2. How Many Assigned Port Numbers?

As noted earlier, systems might require a single port number assignment, but rarely require multiple port numbers. There are a variety of known ways to reduce assigned port number consumption. Although some may be cumbersome or inefficient, they are nearly always preferable to consuming additional port number assignments.

Such techniques include:

- o Use of a discovery service, either a shared service (mDNS), or a discovery service for a given system [RFC6762] [RFC6763].

- o Multiplex packet types using in-band information, either on a per-message or per-connection basis. Such demultiplexing can even hand-off different messages and connections among different processes, such as is done with FTP [RFC959].

There are some cases where NAT and firewall traversal are significantly improved by having an assigned port number. Although

NAT traversal protocols supporting automatic configuration have been proposed and developed (e.g., STUN [RFC5389], TURN [RFC5766], and ICE [RFC5245]), not all application and service designers can rely on their presence as of yet.

In the past, some services were assigned multiple port numbers or sometimes fairly large port ranges (e.g., X11). This occurred for a variety of reasons: port number conservation was not as widely appreciated, assignments were not as ardently reviewed, etc. This no longer reflects current practice and such assignments are not considered to constitute a precedent for future assignments.

7.3. Picking an Assigned Port Number

Given a demonstrated need for a port number assignment, the next question is how to pick the desired port number. An application for a port number assignment does not need to include a desired port number; in that case, IANA will select from those currently available.

Users should consider whether the requested port number is important. For example, would an assignment be acceptable if IANA picked the port number value? Would a TCP (or other transport protocol) port number assignment be useful by itself? If so, a port number can be assigned to a service for one transport protocol where it is already (or can be subsequently) assigned to a different service for other transport protocols.

The most critical issue in picking a number is selecting the desired range, i.e., System vs. User port numbers. The distinction was intended to indicate a difference in privilege; originally, System port numbers required privileged (’root’) access, while User port numbers did not. That distinction has since blurred because some current systems do not limit access control to System port numbers and because some System services have been replicated on User numbers (e.g., IRC). Even so, System port number assignments have continued at an average rate of 3-4 per year over the past 7 years (2007-2013), indicating that the desire to keep this distinction continues.

As a result, the difference between System and User port numbers needs to be treated with caution. Developers are advised to treat services as if they are always run without privilege.

Even when developers seek a System port number assignment, it may be very difficult to obtain. System port number assignment requires IETF Review or IESG Approval and justification that both User and

Dynamic port number ranges are insufficient [RFC6335]. Thus this document recommends both:

>> Developers SHOULD NOT apply for System port number assignments because the increased privilege they are intended to provide is not always enforced.

>> System implementers SHOULD enforce the need for privilege for processes to listen on System port numbers.

At some future date, it might be useful to deprecate the distinction between System and User port numbers altogether. Services typically require elevated (’root’) privileges to bind to a System port number, but many such services go to great lengths to immediately drop those privileges just after connection or other association establishment to reduce the impact of an attack using their capabilities. Such services might be more securely operated on User port numbers than on System port numbers. Further, if System port numbers were no longer assigned, as of 2014 it would cost only 180 of the 1024 System values (17%), or 180 of the overall 49152 assigned (System and User) values (<0.04%).

7.4. Support for Security

Just as a service is a way to obtain information or processing from a host over a network, a service can also be the opening through which to compromise that host. Protecting a service involves security, which includes integrity protection, source authentication, privacy, or any combination of these capabilities. Security can be provided in a number of ways, and thus:

>> New services SHOULD support security capabilities, either directly or via a content protection such as TLS [RFC5246] or DTLS [RFC6347] or transport protection such as TCP-AO [RFC5925]. Insecure versions of new or existing secure services SHOULD be avoided because of the new vulnerability they create.

Secure versions of legacy services that are not already security-capable via in-band negotiations can be very useful. However, there is no IETF consensus on when separate ports should be used for secure and insecure variants of the same service [RFC2595] [RFC2817] [RFC6335]. The overall preference is for use of a single port, as noted in Section 6 of this document and Section 7.2 of [RFC6335], but the appropriate approach depends on the specific characteristics of the service. As a result:

>> When requesting both secure and insecure port assignments for the same service, justification is expected for the utility and safety of each port as an independent service (Section 6). Precedent (e.g., citing other protocols that use a separate insecure port) is inadequate justification by itself.

It’s also important to recognize that port number assignment is not itself a guarantee that traffic using that number provides the corresponding service, or that a given service is always offered only on its assigned port number. Port numbers are ultimately meaningful only between endpoints and any service can be run on any port. Thus:

>> Security SHOULD NOT rely on assigned port number distinctions alone; every service, whether secure or not, is likely to be attacked.

Applications for a new service that requires both a secure and insecure port may be found, on expert review, to be unacceptable, and may not be approved for allocation. Similarly, an application for a new port to support an insecure variant of an existing secure protocol may be found unacceptable. In both cases, the resulting security of the service in practice will be a significant consideration in the decision as to whether to assign an insecure port.

7.5. Support for Future Versions

Requests for assigned port numbers are expected to support multiple versions on the same assigned port number [RFC6335]. Versions are typically indicated in-band, either at the beginning of a connection or other association, or in each protocol message.

>> Version support SHOULD be included in new services rather than relying on different port number assignments for different versions.

>> Version numbers SHOULD NOT be included in either the service name or service description, to avoid the need to make additional port number assignments for future variants of a service.

Again, the assigned port number space is far too limited to be used as an indicator of protocol version or message type. Although this has happened in the past (e.g., for NFS), it should be avoided in new requests.

7.6. Transport Protocols

IANA assigns port numbers specific to one or more transport protocols, typically UDP [RFC768] and TCP [RFC793], but also SCTP [RFC4960], DCCP [RFC4340], and any other standard transport protocol. Originally, IANA port number assignments were concurrent for both UDP and TCP, and other transports were not indicated. However, to conserve the assigned port number space and to reflect increasing use of other transports, assignments are now specific only to the transport being used.

In general, a service should request assignments for multiple transports using the same service name and description on the same port number only when they all reflect essentially the same service. Good examples of such use are DNS and NFS, where the difference between the UDP and TCP services are specific to supporting each transport. E.g., the UDP variant of a service might add sequence numbers and the TCP variant of the same service might add in-band message delimiters. This document does not describe the appropriate selection of a transport protocol for a service.

>> Service names and descriptions for multiple transport port number assignments SHOULD match only when they describe the same service, excepting only enhancements for each supported transport.

When the services differ, it may be acceptable or preferable to use the same port number, but the service names and descriptions should be different for each transport/service pair, reflecting the differences in the services. E.g., if TCP is used for the basic control protocol and UDP for an alarm protocol, then the services might be "name-ctl" and "name-alarm". A common example is when TCP is used for a service and UDP is used to determine whether that service is active (e.g., via a unicast, broadcast, or multicast test message) [RFC1122]. IANA has, for several years, used the suffix "-disc" in service names to distinguish discovery services, such as are used to identify endpoints capable of a given service:

>> Names of discovery services SHOULD use an identifiable suffix; the suggestion is "-disc".

Some services are used for discovery, either in conjunction with a TCP service or as a stand-alone capability. Such services will be more reliable when using multicast rather than broadcast (over IPv4) because IP routers do not forward "all nodes" broadcasts (all 1’s, i.e., 255.255.255.255 for IPv4) and have not been required to support subnet-directed broadcasts since 1999 [RFC1812] [RFC2644].

This issue is relevant only for IPv4 because IPv6 does not support broadcast.

>> UDP over IPv4 multi-host services SHOULD use multicast rather than broadcast.

Designers should be very careful in creating services over transports that do not support congestion control or error recovery, notably UDP. There are several issues that should be considered in such cases, as summarized in Table 1 in [RFC5405]. In addition, the following recommendations apply to service design:

>> Services that use multipoint communication SHOULD be scalable, and SHOULD NOT rely solely on the efficiency of multicast transmission for scalability.

>> Services SHOULD NOT use UDP as a performance enhancement over TCP, e.g., to circumnavigate TCP’s congestion control.

7.7. When to Request an Assignment

Assignments are typically requested when a user has enough information to reasonably answer the questions in the IANA application. IANA applications typically take up to a few weeks to process, with some complex cases taking up to a month. The process typically involves a few exchanges between the IANA Ports Expert Review team and the applicant.

An application needs to include a description of the service, as well as to address key questions designed to help IANA determine whether the assignment is justified. The application should be complete and not refer solely to the Internet Draft, RFC, a website, or any other external documentation.

Services that are independently developed can be requested at any time, but are typically best requested in the last stages of design and initial experimentation, before any deployment has occurred that cannot easily be updated.

>> Users MUST NOT deploy implementations that use assigned port numbers prior their assignment by IANA.

>> Users MUST NOT deploy implementations that default to using the experimental System port numbers (1021 and 1022 [RFC4727]) outside a controlled environment where they can be updated with a subsequent assigned port [RFC3692].

Deployments that use unassigned port numbers before assignment complicate IANA management of the port number space. Keep in mind that this recommendation protects existing assignees, users of current services, and applicants for new assignments; it helps ensure that a desired number and service name are available when assigned. The list of currently unassigned numbers is just that - *currently* unassigned. It does not reflect pending applications. Waiting for an official IANA assignment reduces the chance that an assignment request will conflict with another deployed service.

Applications made through Internet Draft / RFC publication (in any stream) typically use a placeholder ("PORTNUM") in the text, and implementations use an experimental port number until a final assignment has been made [RFC6335]. That assignment is initially indicated in the IANA Considerations section of the document, which is tracked by the RFC Editor. When a document has been approved for publication, that request is forwarded to IANA for handling. IANA will make the new assignment accordingly. At that time, IANA may also request that the applicant fill out the application form on their website, e.g., when the RFC does not directly address the information expected as per [RFC6335]. "Early" assignments can be made when justified, e.g., for early interoperability testing, according to existing process [RFC7120] [RFC6335].

>> Users writing specifications SHOULD use symbolic names for port numbers and service names until an IANA assignment has been completed. Implementations SHOULD use experimental port numbers during this time, but those numbers MUST NOT be cited in documentation except as interim.

7.8. Squatting

"Squatting" describes the use of a number from the assignable range in deployed software without IANA assignment for that use, regardless of whether the number has been assigned or remains available for assignment. It is hazardous because IANA cannot track such usage and thus cannot avoid making legitimate assignments that conflict with such unauthorized usage.

Such "squatted" port numbers remain unassigned, and IANA retains the right to assign them when requested by other applicants. Application and service designers are reminded that is never appropriate to use port numbers that have not been directly assigned [RFC6335]. In particular, any unassigned code from the assigned ranges will be assigned by IANA, and any conflict will be easily resolved as the protocol designer’s fault once that happens (because they would not be the assignee). This may reflect in the public’s judgment on the

quality of their expertise and cooperation with the Internet community.

Regardless, there are numerous services that have squatted on such numbers that are in widespread use. Designers who are using such port numbers are encouraged to apply for an assignment. Note that even widespread de-facto use may not justify a later IANA assignment of that value, especially if either the value has already been assigned to a legitimate applicant or if the service would not qualify for an assignment of its own accord.

7.9. Other Considerations

As noted earlier, System port numbers should be used sparingly, and it is better to avoid them altogether. This avoids the potentially incorrect assumption that the service on such port numbers run in a privileged mode.

Assigned port numbers are not intended to be changed; this includes the corresponding service name. Once deployed, it can be very difficult to recall every implementation, so the assignment should be retained. However, in cases where the current assignee of a name or number has reasonable knowledge of the impact on such uses, and is willing to accept that impact, the name or number of an assignment can be changed [RFC6335]

Aliases, or multiple service names for the same assigned port number, are no longer considered appropriate [RFC6335].

8. Security Considerations

This document focuses on the issues arising when designing services that require new port assignments. Section 7.4 addresses the security and security-related issues of that interaction.

When designing a secure service, the use of TLS [RFC5246], DTLS [RFC6347], or TCP-AO [RFC5925] mechanisms that protect transport protocols or their contents is encouraged. It may not be possible to use IPsec [RFC4301] in similar ways because of the different relationship between IPsec and port numbers and because applications may not be aware of IPsec protections.

This document reminds application and service designers that port numbers do not protect against denial of service attack or guarantee that traffic should be trusted. Using assigned numbers for port filtering isn’t a substitute for authentication, encryption, and integrity protection. The port number alone should not be used to

avoid denial of service attacks or to manage firewall traffic because the use of port numbers is not regulated or validated.

The use of assigned port numbers is the antithesis of privacy because they are intended to explicitly indicate the desired application or service. Strictly, port numbers are meaningful only at the endpoints, so any interpretation elsewhere in the network can be arbitrarily incorrect. However, those numbers can also expose information about available services on a given host. This information can be used by intermediate devices to monitor and intercept traffic as well as to potentially identify key endpoint software properties ("fingerprinting"), which can be used to direct other attacks.

9. IANA Considerations

The entirety of this document focuses on suggestions that help ensure the conservation of port numbers and provide useful hints for issuing informative requests thereof.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2780] Bradner, S., and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.

[RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3962, Jan. 2004.

[RFC4727] Fenner, B., "Experimental Values in IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, November 2006.

[RFC5246] Dierks, T., and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5405] Eggert, L., and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, Nov. 2008.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

[RFC6335] Cotton, M., L. Eggert, J. Touch, M. Westerlund, and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

[RFC6347] Rescorla, E., and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

10.2. Informative References

[IEN112] Postel, J., "Transmission Control Protocol", IEN 112, August 1979.

[RFC33] Crocker, S., "New Host-Host Protocol", RFC 33 February 1970.

[RFC37] Crocker, S., "Network Meeting Epilogue", RFC 37, March 1970.

[RFC38] Wolfe, S., "Comments on Network Protocol from NWG/RFC #36", RFC 38, March 1970.

[RFC48] Postel, J., and S. Crocker, "Possible protocol plateau", RFC 48, April 1970.

[RFC61] Walden, D., "Note on Interprocess Communication in a Resource Sharing Computer Network", RFC 61, July 1970.

[RFC76] Bouknight, J., J. Madden, and G. Grossman, "Connection by name: User oriented protocol", RFC 76, October 1970.

[RFC333] Bressler, R., D. Murphy, and D. Walden. "Proposed experiment with a Message Switching Protocol", RFC 333, May 1972.

[RFC739] Postel, J., "Assigned numbers", RFC 739, November 1977.

[RFC758] Postel, J., "Assigned numbers", RFC 758, August 1979.

[RFC768] Postel, J., "User Datagram Protocol", RFC 768, August 1980.

[RFC793] Postel, J., "Transmission Control Protocol" RFC 793, September 1981

[RFC820] Postel, J., "Assigned numbers", RFC 820, August 1982.

[RFC900] Reynolds, J., and J. Postel, "Assigned numbers", RFC 900, June 1984.

[RFC959] Postel, J., and J. Reynolds, "FILE TRANSFER PROTOCOL (FTP)", RFC 959, October 1985.

[RFC1122] Braden, B. (Ed.), "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989.

[RFC1340] Reynolds, J., and J. Postel, "Assigned numbers", RFC 1340, July 1992.

[RFC1700] Reynolds, J., and J. Postel, "Assigned numbers", RFC 1700, October 1994.

[RFC1812] Baker, F. (Ed.), "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2", RFC 1833, August 1995.

[RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999.

[RFC2644] Senie, D., "Changing the Default for Directed Broadcasts in Routers", RFC 2644, August 1999.

[RFC2817] Khare, R., and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.

[RFC3232] Reynolds, J. (Ed.), "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, January 2002.

[RFC3261] Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC4301] Kent, S., and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4340] Kohler, E., M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4960] Stewart, R. (Ed.), "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5389] Rosenberg, J., R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT", RFC 5389, October 2008.

[RFC5766] Mahy, R., P. Matthews, and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6762] Cheshire, S., and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[RFC6763] Cheshire, S., and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, January 2014.

[RFC7230] Fielding, R., (Ed.), and J. Reshke, (Ed.), "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

11. Acknowledgments

This work benefitted from the feedback from David Black, Lars Eggert, Gorry Fairhurst, and Eliot Lear, as well as discussions of the IETF TSVWG WG.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292-6695
U.S.A.

Phone: +1 (310) 448-9151
EMail: touch@isi.edu

Transport Area Working Group
Internet-Draft
Obsoletes: 5405 (if approved)
Intended status: Best Current Practice
Expires: April 17, 2017

L. Eggert
NetApp
G. Fairhurst
University of Aberdeen
G. Shepherd
Cisco Systems

October 14, 2016

UDP Usage Guidelines
draft-ietf-tsvwg-rfc5405bis-19

Abstract

The User Datagram Protocol (UDP) provides a minimal message-passing transport that has no inherent congestion control mechanisms. This document provides guidelines on the use of UDP for the designers of applications, tunnels and other protocols that use UDP. Congestion control guidelines are a primary focus, but the document also provides guidance on other topics, including message sizes, reliability, checksums, middlebox traversal, the use of ECN, DSCPs, and ports.

Because congestion control is critical to the stable operation of the Internet, applications and other protocols that choose to use UDP as an Internet transport must employ mechanisms to prevent congestion collapse and to establish some degree of fairness with concurrent traffic. They may also need to implement additional mechanisms, depending on how they use UDP.

Some guidance is also applicable to the design of other protocols (e.g., protocols layered directly on IP or via IP-based tunnels), especially when these protocols do not themselves provide congestion control.

This document obsoletes RFC5405 and adds guidelines for multicast UDP usage.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The User Datagram Protocol (UDP) [RFC0768] provides a minimal, unreliable, best-effort, message-passing transport to applications and other protocols (such as tunnels) that desire to operate over IP. Both are simply called "applications" in the remainder of this document.

Compared to other transport protocols, UDP and its UDP-Lite variant [RFC3828] are unique in that they do not establish end-to-end connections between communicating end systems. UDP communication consequently does not incur connection establishment and teardown overheads, and there is minimal associated end system state. Because of these characteristics, UDP can offer a very efficient communication transport to some applications.

A second unique characteristic of UDP is that it provides no inherent congestion control mechanisms. On many platforms, applications can send UDP datagrams at the line rate of the platform's link interface, which is often much greater than the available end-to-end path capacity, and doing so contributes to congestion along the path. [RFC2914] describes the best current practice for congestion control in the Internet. It identifies two major reasons why congestion control mechanisms are critical for the stable operation of the Internet:

1. The prevention of congestion collapse, i.e., a state where an increase in network load results in a decrease in useful work done by the network.

2. The establishment of a degree of fairness, i.e., allowing multiple flows to share the capacity of a path reasonably equitably.

Because UDP itself provides no congestion control mechanisms, it is up to the applications that use UDP for Internet communication to employ suitable mechanisms to prevent congestion collapse and establish a degree of fairness. [RFC2309] discusses the dangers of congestion-unresponsive flows and states that "all UDP-based streaming applications should incorporate effective congestion avoidance mechanisms." [RFC7567] reaffirms this statement. This is an important requirement, even for applications that do not use UDP for streaming. In addition, congestion-controlled transmission is of benefit to an application itself, because it can reduce self-induced packet loss, minimize retransmissions, and hence reduce delays. Congestion control is essential even at relatively slow transmission

rates. For example, an application that generates five 1500-byte UDP datagrams in one second can already exceed the capacity of a 56 Kb/s path. For applications that can operate at higher, potentially unbounded data rates, congestion control becomes vital to prevent congestion collapse and establish some degree of fairness. Section 3 describes a number of simple guidelines for the designers of such applications.

A UDP datagram is carried in a single IP packet and is hence limited to a maximum payload of 65,507 bytes for IPv4 and 65,527 bytes for IPv6. The transmission of large IP packets usually requires IP fragmentation. Fragmentation decreases communication reliability and efficiency and should be avoided. IPv6 allows the option of transmitting large packets ("jumbograms") without fragmentation when all link layers along the path support this [RFC2675]. Some of the guidelines in Section 3 describe how applications should determine appropriate message sizes. Other sections of this document provide guidance on reliability, checksums, middlebox traversal and use of multicast.

This document provides guidelines and recommendations. Although most UDP applications are expected to follow these guidelines, there do exist valid reasons why a specific application may decide not to follow a given guideline. In such cases, it is RECOMMENDED that application designers cite the respective section(s) of this document in the technical specification of their application or protocol and explain their rationale for their design choice.

[RFC5405] was scoped to provide guidelines for unicast applications only, whereas this document also provides guidelines for UDP flows that use IP anycast, multicast, broadcast, and applications that use UDP tunnels to support IP flows.

Finally, although this document specifically refers to usage of UDP, the spirit of some of its guidelines also applies to other message-passing applications and protocols (specifically on the topics of congestion control, message sizes, and reliability). Examples include signaling, tunnel, or control applications that choose to run directly over IP by registering their own IP protocol number with IANA. This document is expected to provide useful background reading to the designers of such applications and protocols.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. UDP Usage Guidelines

Internet paths can have widely varying characteristics, including transmission delays, available bandwidths, congestion levels, reordering probabilities, supported message sizes, or loss rates. Furthermore, the same Internet path can have very different conditions over time. Consequently, applications that may be used on the Internet MUST NOT make assumptions about specific path characteristics. They MUST instead use mechanisms that let them operate safely under very different path conditions. Typically, this requires conservatively probing the current conditions of the Internet path they communicate over to establish a transmission behavior that it can sustain and that is reasonably fair to other traffic sharing the path.

These mechanisms are difficult to implement correctly. For most applications, the use of one of the existing IETF transport protocols is the simplest method of acquiring the required mechanisms. Doing so also avoids issues that protocols using a new IP protocol number face when being deployed over the Internet, where middleboxes that only support TCP and UDP are sometimes present. Consequently, the RECOMMENDED alternative to the UDP usage described in the remainder of this section is the use of an IETF transport protocol such as TCP [RFC0793], Stream Control Transmission Protocol (SCTP) [RFC4960], and SCTP Partial Reliability Extension (SCTP-PR) [RFC3758], or Datagram Congestion Control Protocol (DCCP) [RFC4340] with its different congestion control types [RFC4341][RFC4342][RFC5622], or transport protocols specified by the IETF in the future. (UDP-encapsulated SCTP [RFC6951] and DCCP [RFC6773] can offer support for traversing firewalls and other middleboxes where the native protocols are not supported.)

If used correctly, these more fully-featured transport protocols are not as "heavyweight" as often claimed. For example, the TCP algorithms have been continuously improved over decades, and have reached a level of efficiency and correctness that custom application-layer mechanisms will struggle to easily duplicate. In addition, many TCP implementations allow connections to be tuned by an application to its purposes. For example, TCP’s "Nagle" algorithm [RFC1122] can be disabled, improving communication latency at the expense of more frequent -- but still congestion-controlled -- packet transmissions. Another example is the TCP SYN cookie mechanism [RFC4987], which is available on many platforms. TCP with SYN cookies does not require a server to maintain per-connection state until the connection is established. TCP also requires the end that closes a connection to maintain the TIME-WAIT state that prevents delayed segments from one connection instance from interfering with a later one. Applications that are aware of and designed for this

behavior can shift maintenance of the TIME-WAIT state to conserve resources by controlling which end closes a TCP connection [FABER]. Finally, TCP's built-in capacity-probing and awareness of the maximum transmission unit supported by the path (PMTU) results in efficient data transmission that quickly compensates for the initial connection setup delay, in the case of transfers that exchange more than a few segments.

3.1. Congestion Control Guidelines

If an application or protocol chooses not to use a congestion-controlled transport protocol, it SHOULD control the rate at which it sends UDP datagrams to a destination host, in order to fulfill the requirements of [RFC2914]. It is important to stress that an application SHOULD perform congestion control over all UDP traffic it sends to a destination, independently from how it generates this traffic. For example, an application that forks multiple worker processes or otherwise uses multiple sockets to generate UDP datagrams SHOULD perform congestion control over the aggregate traffic.

Several approaches to perform congestion control are discussed in the remainder of this section. The section describes generic topics with an intended emphasis on unicast and anycast [RFC1546] usage. Not all approaches discussed below are appropriate for all UDP-transmitting applications. Section 3.1.2 discusses congestion control options for applications that perform bulk transfers over UDP. Such applications can employ schemes that sample the path over several subsequent round-trips during which data is exchanged to determine a sending rate that the path at its current load can support. Other applications only exchange a few UDP datagrams with a destination. Section 3.1.3 discusses congestion control options for such "low data-volume" applications. Because they typically do not transmit enough data to iteratively sample the path to determine a safe sending rate, they need to employ different kinds of congestion control mechanisms. Section 3.1.11 discusses congestion control considerations when UDP is used as a tunneling protocol. Section 4 provides additional recommendations for broadcast and multicast usage.

It is important to note that congestion control should not be viewed as an add-on to a finished application. Many of the mechanisms discussed in the guidelines below require application support to operate correctly. Application designers need to consider congestion control throughout the design of their application, similar to how they consider security aspects throughout the design process.

In the past, the IETF has also investigated integrated congestion control mechanisms that act on the traffic aggregate between two hosts, i.e., a framework such as the Congestion Manager [RFC3124], where active sessions may share current congestion information in a way that is independent of the transport protocol. Such mechanisms have currently failed to see deployment, but would otherwise simplify the design of congestion control mechanisms for UDP sessions, so that they fulfill the requirements in [RFC2914].

3.1.1. Protocol Timer Guidelines

Understanding the latency between communicating endpoints is usually a crucial part of effective congestion control implementations for protocols and applications. Latency estimation can be used in a number of protocol functions, such as calculating a congestion-controlled transmission rate, triggering retransmission, and detecting packet loss. Additional protocol functions, for example, determining an interval for probing a path, determining an interval between keep-alive messages, determining an interval for measuring the quality of experience, or determining if a remote endpoint has responded to a request to perform an action typically operate over longer timescales than congestion control and therefore are not covered in this section.

The general recommendation in this document is that applications SHOULD leverage existing congestion control techniques and the latency estimators specified therein (see next subsection). The following guidelines are provided for applications that need to design their own latency estimation mechanisms.

The guidelines are framed in terms of "latency" and not "round-trip time" because some situations require characterizing only the network-based latency (e.g., TCP-Friendly Rate Control [RFC5348]), while other cases necessitate inclusion of the time required by the remote endpoint to provide feedback (e.g., developing an understanding of when to retransmit a message).

The latency between endpoints is generally a dynamic property. Therefore, estimates SHOULD represent some sort of averaging of multiple recent measurement samples to account for variance. Leveraging an Exponentially Weighted Moving Average (EWMA) has proven useful for this purpose (e.g., in TCP [RFC6298] and TCP-Friendly Rate Control (TFRC) [RFC5348]).

Independent latency estimates SHOULD be maintained for each destination with which an endpoint communicates.

Latency samples MUST NOT be derived from ambiguous transactions. The canonical example is in a protocol that retransmits data, but subsequently cannot determine which copy is being acknowledged. This ambiguity makes correct computation of the latency problematic. See the discussion of Karn's algorithm in [RFC6298]. This requirement ensures a sender establishes a sound estimate of the latency without relying on mis-leading measurements.

When a latency estimate is used to arm a timer that provides loss detection - with or without retransmission - expiry of the timer MUST be interpreted as an indication of congestion in the network, causing the sending rate to be adapted to a safe conservative rate (e.g., TCP collapses the congestion window to one segment [RFC5681]).

Some applications require an initial latency estimate before the latency between endpoints can be empirically sampled. For instance, when arming a retransmission timer an initial value is needed to protect the messages sent before the endpoints sample the latency. This initial latency estimate SHOULD generally be as conservative (large) as possible for the given application. For instance, in the absence of any knowledge about the latency of a path, TCP requires the initial Retransmission Timeout (RTO) to be set to no less than 1 second [RFC6298]. UDP applications SHOULD similarly use an initial latency estimate of 1 second. Values shorter than 1 second can be problematic (see the data analysis in the appendix of [RFC6298]).

3.1.2. Bulk Transfer Applications

Applications that perform bulk transmission of data to a peer over UDP, i.e., applications that exchange more than a few UDP datagrams per round-trip time (RTT), SHOULD implement TFRC [RFC5348], window-based TCP-like congestion control, or otherwise ensure that the application complies with the congestion control principles.

TFRC has been designed to provide both congestion control and fairness in a way that is compatible with the IETF's other transport protocols. If an application implements TFRC, it need not follow the remaining guidelines in Section 3.1.2, because TFRC already addresses them, but SHOULD still follow the remaining guidelines in the subsequent subsections of Section 3.

Bulk transfer applications that choose not to implement TFRC or TCP-like windowing SHOULD implement a congestion control scheme that results in bandwidth (capacity) use that competes fairly with TCP within an order of magnitude.

Section 2 of [RFC3551] suggests that applications SHOULD monitor the packet loss rate to ensure that it is within acceptable parameters.

Packet loss is considered acceptable if a TCP flow across the same network path under the same network conditions would achieve an average throughput, measured on a reasonable timescale, that is not less than that of the UDP flow. The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in timescale and throughput. The recommendations for managing timers specified in Section 3.1.1 also apply.

Finally, some bulk transfer applications may choose not to implement any congestion control mechanism and instead rely on transmitting across reserved path capacity (see Section 3.1.9). This might be an acceptable choice for a subset of restricted networking environments, but is by no means a safe practice for operation over the wider Internet. When the UDP traffic of such applications leaks out into unprovisioned Internet paths, it can significantly degrade the performance of other traffic sharing the path and even result in congestion collapse. Applications that support an uncontrolled or unadaptive transmission behavior SHOULD NOT do so by default and SHOULD instead require users to explicitly enable this mode of operation, and they SHOULD verify that sufficient path capacity has been reserved for them.

3.1.3. Low Data-Volume Applications

When applications that at any time exchange only a few UDP datagrams with a destination implement TFRC or one of the other congestion control schemes in Section 3.1.2, the network sees little benefit, because those mechanisms perform congestion control in a way that is only effective for longer transmissions.

Applications that at any time exchange only a few UDP datagrams with a destination SHOULD still control their transmission behavior by not sending on average more than one UDP datagram per RTT to a destination. Similar to the recommendation in [RFC1536], an application SHOULD maintain an estimate of the RTT for any destination with which it communicates using the methods specified in Section 3.1.1.

Some applications cannot maintain a reliable RTT estimate for a destination. These applications do not need to or are unable to use protocol timers to measure the RTT (Section 3.1.1). Two cases can be identified:

1. The first case is that of applications that exchange too few UDP datagrams with a peer to establish a statistically accurate RTT estimate, but can monitor the reliability of transmission (Section 3.3). Such applications MAY use a predetermined transmission interval that is exponentially backed-off when

packets are found to be lost. TCP specifies an initial value of 1 second [RFC6298], which is also RECOMMENDED as an initial value for UDP applications. Some low data-volume applications, e.g., SIP [RFC3261] and GIST [RFC5971] use an interval of 500 ms, and shorter values are likely problematic in many cases. As in the previous case, note that the initial timeout is not the maximum possible timeout, see Section 3.1.1.

2. A second case of applications cannot maintain an RTT estimate for a destination, because the destination does not send return traffic. Such applications SHOULD NOT send more than one UDP datagram every 3 seconds, and SHOULD use an even less aggressive rate when possible. Shorter values are likely problematic in many cases. Note that the sending rate in this case must be more conservative than in the previous cases, because the lack of return traffic prevents the detection of packet loss, i.e., congestion, and the application therefore cannot perform exponential back-off to reduce load.

3.1.4. Applications supporting bidirectional communications

Applications that communicate bidirectionally SHOULD employ congestion control for both directions of the communication. For example, for a client-server, request-response-style application, clients SHOULD congestion-control their request transmission to a server, and the server SHOULD congestion-control its responses to the clients. Congestion in the forward and reverse direction is uncorrelated, and an application SHOULD either independently detect and respond to congestion along both directions, or limit new and retransmitted requests based on acknowledged responses across the entire round-trip path.

3.1.5. Implications of RTT and Loss Measurements on Congestion Control

Transports such as TCP, SCTP and DCCP provide timely detection of congestion that results in an immediate reduction of their maximum sending rate when congestion is experienced. This reaction is typically completed 1-2 RTTs after loss/congestion is encountered. Applications using UDP SHOULD implement a congestion control scheme that provides a prompt reaction to signals indicating congestion (e.g., by reducing the rate within the next RTT following a congestion signal).

The operation of a UDP congestion control algorithm can be very different to the way TCP operates. This includes congestion controls that respond on timescales that fit applications that cannot usefully work within the "change rate every RTT" model of TCP. Applications that experience a low or varying RTT are particularly vulnerable to

sampling errors (e.g., due to measurement noise, or timer accuracy). This suggests the need to average loss/congestion and RTT measurements over a longer interval, however this also can contribute additional delay in detecting congestion. Some applications may not react by reducing their sending rate immediately for various reasons, including: RTT and loss measurements are only made periodically (e.g., using RTCP), additional time is required to filter information, or the application is only able to change its sending rate at predetermined interval (e.g., some video codecs).

When designing a congestion control algorithm, the designer therefore needs to consider the total time taken to reduce the load following a lack of feedback or a congestion event. An application where the most recent RTT measurement is smaller than the actual RTT or the measured loss rate is smaller than the current rate, can result in over estimating the available capacity. Such over estimation can result in a sending rate that creates congestion to the application or other flows sharing the path capacity, and can contribute to congestion collapse - both of these need to be avoided.

A congestion control designed for UDP SHOULD respond as quickly as possible when it experiences congestion, and SHOULD take into account both the loss rate and the response time when choosing a new rate. The implemented congestion control scheme SHOULD result in bandwidth (capacity) use that is comparable to that of TCP within an order of magnitude, so that it does not starve other flows sharing a common bottleneck.

3.1.6. Burst Mitigation and Pacing

UDP applications SHOULD provide mechanisms to regulate the bursts of transmission that the application may send to the network. Many TCP and SCTP implementations provide mechanisms that prevent a sender from generating long bursts at line-rate, since these are known to induce early loss to applications sharing a common network bottleneck. The use of pacing with TCP [ALLMAN] has also been shown to improve the coexistence of TCP flows with other flows. The need to avoid excessive transmission bursts is also noted in specifications for applications (e.g., [RFC7143]).

Even low data-volume UDP flows may benefit from packet pacing, e.g., an application that sends three copies of a packet to improve robustness to loss is RECOMMENDED to pace out those three packets over several RTTs, to reduce the probability that all three packets will be lost due to the same congestion event (or other event, such as burst corruption).

3.1.7. Explicit Congestion Notification

Internet applications can use Explicit Congestion Notification (ECN) [RFC3168] to gain benefits for the services they support [I-D.ietf-aqm-ecn-benefits].

Internet transports, such as TCP, provide a set of mechanisms that are needed to utilize ECN. ECN operates by setting an ECN-capable codepoint (ECT(0) or ECT(1)) in the IP header of packets that are sent. This indicates to ECN-capable network devices (routers, and other devices) that they may mark (set the congestion experienced, CE codepoint), rather than drop the IP packet as a signal of incipient congestion.

UDP applications can also benefit from enabling ECN, providing that the API supports ECN and that they implement the required protocol mechanisms to support ECN.

The set of mechanisms required for an application to use ECN over UDP are:

- o A sender MUST provide a method to determine (e.g., negotiate) that the corresponding application is able to provide ECN feedback using a compatible ECN method.

- o A receiver that enables the use of ECN for a UDP port MUST check the ECN field at the receiver for each UDP datagram that it receives on this port.

- o The receiving application needs to provide feedback of congestion information to the sending application. This MUST report the presence of datagrams received with a CE-mark by providing a mechanism to feed this congestion information back to the sending application. The feedback MAY also report the presence of ECT(1) and ECT(0)/Not-ECT packets [RFC7560]. ([RFC3168] and [RFC7560] specify methods for TCP.)

- o An application sending ECN-capable datagrams MUST provide an appropriate congestion reaction when it receives feedback indicating that congestion has been experienced. This ought to result in reduction of the sending rate by the UDP congestion control method (see Section 3.1) that is not less than the reaction of TCP under equivalent conditions.

- o A sender SHOULD detect network paths that do not support the ECN field correctly. When detected they need to either conservatively react to congestion or even fall back to not using ECN [I-D.ietf-aqm-ecn-benefits]. This method needs to be robust to

changes within the network path that may occur over the lifetime of a session.

- o A sender is encouraged to provide a mechanism to detect and react appropriately to misbehaving receivers that fail to report CE-marked packets [I-D.ietf-aqm-ecn-benefits].

[RFC6679] provides guidance and an example of this support, by describing a method to allow ECN to be used for UDP-based applications using the Real-Time Protocol (RTP). Applications that cannot provide this set of mechanisms, but wish to gain the benefits of using ECN, are encouraged to use a transport protocol that already supports ECN (such as TCP).

3.1.8. Differentiated Services Model

An application using UDP can use the differentiated services (DiffServ) Quality of Service (QoS) framework. To enable differentiated services processing, a UDP sender sets the Differentiated Services Code Point (DSCP) field [RFC2475] in packets sent to the network. Normally, a UDP source/destination port pair will set a single DSCP value for all packets belonging to a flow, but multiple DSCPs can be used as described later in this section. A DSCP may be chosen from a small set of fixed values (the class selector code points), or from a set of recommended values defined in the Per Hop Behavior (PHB) specifications, or from values that have purely local meanings to a specific network that supports DiffServ. In general, packets may be forwarded across multiple networks between source and destination.

In setting a non-default DSCP value, an application must be aware that DSCP markings may be changed or removed between the traffic source and destination. This has implications on the design of applications that use DSCPs. Specifically, applications SHOULD be designed to not rely on implementation of a specific network treatment, they need instead to implement congestion control methods to determine if their current sending rate is inducing congestion in the network.

[RFC7657] describes the implications of using DSCPs and provides recommendations on using multiple DSCPs within a single network five-tuple (source and destination addresses, source and destination ports, and the transport protocol used, in this case, UDP or UDP-Lite), and particularly the expected impact on transport protocol interactions, with congestion control or reliability functionality (e.g., retransmission, reordering). Use of multiple DSCPs can result in reordering by increasing the set of network forwarding resources

used by a sender. It can also increase exposure to resource depletion or failure.

3.1.9. QoS, Pre-Provisioned or Reserved Capacity

The IETF usually specifies protocols for use within the Best Effort General Internet. Sometimes it is relevant to specify protocols with a different applicability. An application using UDP can use the integrated services QoS framework. This framework is usually made available within controlled environments (e.g., within a single administrative domain or bilaterally agreed connection between domains). Applications intended for the Internet SHOULD NOT assume that QoS mechanisms are supported by the networks they use, and therefore need to provide congestion control, error recovery, etc. in case the actual network path does not provide provisioned service.

Some UDP applications are only expected to be deployed over network paths that use pre-provisioned capacity or capacity reserved using dynamic provisioning, e.g., through the Resource Reservation Protocol (RSVP). Multicast applications are also used with pre-provisioned capacity (e.g., IPTV deployments within access networks). These applications MAY choose not to implement any congestion control mechanism and instead rely on transmitting only on paths where the capacity is provisioned and reserved for this use. This might be an acceptable choice for a subset of restricted networking environments, but is by no means a safe practice for operation over the wider Internet. Applications that choose this option SHOULD carefully and in detail describe the provisioning and management procedures that result in the desired containment.

Applications that support an uncontrolled or unadaptive transmission behavior SHOULD NOT do so by default and SHOULD instead require users to explicitly enable this mode of operation.

Applications designed for use within a controlled environment (see Section 3.6) may be able to exploit network management functions to detect whether they are causing congestion, and react accordingly. If the traffic of such applications leaks out into unprovisioned Internet paths, it can significantly degrade the performance of other traffic sharing the path and even result in congestion collapse. Protocols designed for such networks SHOULD provide mechanisms at the network edge to prevent leakage of traffic into unprovisioned Internet paths (e.g., [RFC7510]). To protect other applications sharing the same path, applications SHOULD also deploy an appropriate circuit breaker, as described in Section 3.1.10.

An IETF specification targeting a controlled environment is expected to provide an applicability statement that restricts the application to the controlled environment (see Section 3.6).

3.1.10. Circuit Breaker Mechanisms

A transport circuit breaker is an automatic mechanism that is used to estimate the congestion caused by a flow, and to terminate (or significantly reduce the rate of) the flow when excessive congestion is detected [I-D.ietf-tsvwg-circuit-breaker]. This is a safety measure to prevent congestion collapse (starvation of resources available to other flows), essential for an Internet that is heterogeneous and for traffic that is hard to predict in advance.

A circuit breaker is intended as a protection mechanism of last resort. Under normal circumstances, a circuit breaker should not be triggered; it is designed to protect things when there is severe overload. The goal is usually to limit the maximum transmission rate that reflects the available capacity of a network path. Circuit breakers can operate on individual UDP flows or traffic aggregates, e.g., traffic sent using a network tunnel.

[I-D.ietf-tsvwg-circuit-breaker] provides guidance and examples on the use of circuit breakers. The use of a circuit breaker in RTP is specified in [I-D.ietf-avtcore-rtp-circuit-breakers].

Applications used in the general Internet SHOULD implement a transport circuit breaker if they do not implement congestion control or operate a low volume data service (see Section 3.6). All applications MAY implement a transport circuit breaker [I-D.ietf-tsvwg-circuit-breaker] and are encouraged to consider implementing at least a slow-acting transport circuit breaker to provide a protection of last resort for their network traffic.

3.1.11. UDP Tunnels

One increasingly popular use of UDP is as a tunneling protocol [I-D.ietf-intarea-tunnels], where a tunnel endpoint encapsulates the packets of another protocol inside UDP datagrams and transmits them to another tunnel endpoint, which decapsulates the UDP datagrams and forwards the original packets contained in the payload. One example of such a protocol is Teredo [RFC4380]. Tunnels establish virtual links that appear to directly connect locations that are distant in the physical Internet topology and can be used to create virtual (private) networks. Using UDP as a tunneling protocol is attractive when the payload protocol is not supported by middleboxes that may exist along the path, because many middleboxes support transmission using UDP.

Well-implemented tunnels are generally invisible to the endpoints that happen to transmit over a path that includes tunneled links. On the other hand, to the routers along the path of a UDP tunnel, i.e., the routers between the two tunnel endpoints, the traffic that a UDP tunnel generates is a regular UDP flow, and the encapsulator and decapsulator appear as regular UDP-sending and -receiving applications. Because other flows can share the path with one or more UDP tunnels, congestion control needs to be considered.

Two factors determine whether a UDP tunnel needs to employ specific congestion control mechanisms -- first, whether the payload traffic is IP-based; second, whether the tunneling scheme generates UDP traffic at a volume that corresponds to the volume of payload traffic carried within the tunnel.

IP-based unicast traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based unicast traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. Consequently, a tunnel carrying IP-based unicast traffic should already interact appropriately with other traffic sharing the path, and specific congestion control mechanisms for the tunnel are not necessary.

However, if the IP traffic in the tunnel is known to not be congestion-controlled, additional measures are RECOMMENDED to limit the impact of the tunneled traffic on other traffic sharing the path. For the specific case of a tunnel that carries IP multicast traffic, see Section 4.1.

The following guidelines define these possible cases in more detail:

1. A tunnel generates UDP traffic at a volume that corresponds to the volume of payload traffic, and the payload traffic is IP-based and congestion-controlled.

   This is arguably the most common case for Internet tunnels. In this case, the UDP tunnel SHOULD NOT employ its own congestion control mechanism, because congestion losses of tunneled traffic will already trigger an appropriate congestion response at the original senders of the tunneled traffic. A circuit breaker mechanism may provide benefit by controlling the envelope of the aggregated traffic.

   Note that this guideline is built on the assumption that most IP-based communication is congestion-controlled. If a UDP tunnel is used for IP-based traffic that is known to not be congestion-controlled, the next set of guidelines applies.

2. A tunnel generates UDP traffic at a volume that corresponds to the volume of payload traffic, and the payload traffic is not known to be IP-based, or is known to be IP-based but not congestion-controlled.

   This can be the case, for example, when some link-layer protocols are encapsulated within UDP (but not all link-layer protocols; some are congestion-controlled). Because it is not known that congestion losses of tunneled non-IP traffic will trigger an appropriate congestion response at the senders, the UDP tunnel SHOULD employ an appropriate congestion control mechanism or circuit breaker mechanism designed for the traffic it carries. Because tunnels are usually bulk-transfer applications as far as the intermediate routers are concerned, the guidelines in Section 3.1.2 apply.

3. A tunnel generates UDP traffic at a volume that does not correspond to the volume of payload traffic, independent of whether the payload traffic is IP-based or congestion-controlled.

   Examples of this class include UDP tunnels that send at a constant rate, increase their transmission rates under loss, for example, due to increasing redundancy when Forward Error Correction is used, or are otherwise unconstrained in their transmission behavior. These specialized uses of UDP for tunneling go beyond the scope of the general guidelines given in this document. The implementer of such specialized tunnels SHOULD carefully consider congestion control in the design of their tunneling mechanism and SHOULD consider use of a circuit breaker mechanism.

The type of encapsulated payload might be identified by a UDP port; identified by an Ethernet Type or IP protocol number. A tunnel SHOULD provide mechanisms to restrict the types of flows that may be carried by the tunnel. For instance, a UDP tunnel designed to carry IP needs to filter out non-IP traffic at the ingress. This is particularly important when a generic tunnel encapsulation is used (e.g., one that encapsulates using an EtherType value). Such tunnels SHOULD provide a mechanism to restrict the types of traffic that are allowed to be encapsulated for a given deployment (see [I-D.ietf-intarea-tunnels]).

Designing a tunneling mechanism requires significantly more expertise than needed for many other UDP applications, because tunnels are usually intended to be transparent to the endpoints transmitting over them, so they need to correctly emulate the behavior of an IP link [I-D.ietf-intarea-tunnels], e.g.:

- o Requirements for tunnels that carry or encapsulate using ECN code points [RFC6040].

- o Usage of the IP DSCP field by tunnel endpoints [RFC2983].

- o Encapsulation considerations in the design of tunnels [I-D.ietf-rtgwg-dt-encap].

- o Usage of ICMP messages [I-D.ietf-intarea-tunnels].

- o Handling of fragmentation and packet size for tunnels [I-D.ietf-intarea-tunnels].

- o Source port usage for tunnels designed to support equal cost multipath (ECMP) routing (see Section 5.1.1).

- o Guidance on the need to protect headers [I-D.ietf-intarea-tunnels] and the use of checksums for IPv6 tunnels (see Section 3.4.1).

- o Support for operations and maintenance [I-D.ietf-intarea-tunnels].

At the same time, the tunneled traffic is application traffic like any other from the perspective of the networks the tunnel transmits over. This document only touches upon the congestion control considerations for implementing UDP tunnels; a discussion of other required tunneling behavior is out of scope.

3.2. Message Size Guidelines

IP fragmentation lowers the efficiency and reliability of Internet communication. The loss of a single fragment results in the loss of an entire fragmented packet, because even if all other fragments are received correctly, the original packet cannot be reassembled and delivered. This fundamental issue with fragmentation exists for both IPv4 and IPv6.

In addition, some network address translators (NATs) and firewalls drop IP fragments. The network address translation performed by a NAT only operates on complete IP packets, and some firewall policies also require inspection of complete IP packets. Even with these being the case, some NATs and firewalls simply do not implement the necessary reassembly functionality, and instead choose to drop all fragments. Finally, [RFC4963] documents other issues specific to IPv4 fragmentation.

Due to these issues, an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmission Unit (MTU) along the path to the destination. Consequently, an

application SHOULD either use the path MTU information provided by the IP layer or implement Path MTU Discovery (PMTUD) itself [RFC1191][RFC1981][RFC4821] to determine whether the path to a destination will support its desired message size without fragmentation. However, the ICMP messages that enable path MTU discovery are being increasingly filtered by middleboxes (including Firewalls) [RFC4890]. When the path includes a tunnel, some devices acting as a tunnel ingress discard ICMP messages that originate from network devices over which the tunnel passes, preventing these reaching the UDP endpoint.

Packetization Layer Path MTU Discovery (PLPMTUD) [RFC4821] does not rely upon network support for ICMP messages and is therefore considered more robust than standard PMTUD. It is not susceptible to "black holing" of ICMP message. To operate, PLPMTUD requires changes to the way the transport is used, both to transmit probe packets, and to account for the loss or success of these probes. This updates not only the PMTU algorithm, it also impacts loss recovery, congestion control, etc. These updated mechanisms can be implemented within a connection-oriented transport (e.g., TCP, SCTP, DCCP), but are not a part of UDP, but this type of feedback is not typically present for unidirectional applications.

PLPMTUD therefore places additional design requirements on a UDP application that wishes to use this method. This is especially true for UDP tunnels, because the overhead of sending probe packets needs to be accounted for and may require adding a congestion control mechanism to the tunnel (see Section 3.1.11) as well as complicating the data path at a tunnel decapsulator.

Applications that do not follow this recommendation to do PMTU/ PLPMTUD discovery SHOULD still avoid sending UDP datagrams that would result in IP packets that exceed the path MTU. Because the actual path MTU is unknown, such applications SHOULD fall back to sending messages that are shorter than the default effective MTU for sending (EMTU_S in [RFC1122]). For IPv4, EMTU_S is the smaller of 576 bytes and the first-hop MTU [RFC1122]. For IPv6, EMTU_S is 1280 bytes [RFC2460]. The effective PMTU for a directly connected destination (with no routers on the path) is the configured interface MTU, which could be less than the maximum link payload size. Transmission of minimum-sized UDP datagrams is inefficient over paths that support a larger PMTU, which is a second reason to implement PMTU discovery.

To determine an appropriate UDP payload size, applications MUST subtract the size of the IP header (which includes any IPv4 optional headers or IPv6 extension headers) as well as the length of the UDP header (8 bytes) from the PMTU size. This size, known as the Maximum Segment Size (MSS), can be obtained from the TCP/IP stack [RFC1122].

Applications that do not send messages that exceed the effective PMTU of IPv4 or IPv6 need not implement any of the above mechanisms. Note that the presence of tunnels can cause an additional reduction of the effective PMTU [I-D.ietf-intarea-tunnels], so implementing PMTU discovery may be beneficial.

Applications that fragment an application-layer message into multiple UDP datagrams SHOULD perform this fragmentation so that each datagram can be received independently, and be independently retransmitted in the case where an application implements its own reliability mechanisms.

3.3. Reliability Guidelines

Application designers are generally aware that UDP does not provide any reliability, e.g., it does not retransmit any lost packets. Often, this is a main reason to consider UDP as a transport protocol. Applications that do require reliable message delivery MUST implement an appropriate mechanism themselves.

UDP also does not protect against datagram duplication, i.e., an application may receive multiple copies of the same UDP datagram, with some duplicates arriving potentially much later than the first. Application designers SHOULD handle such datagram duplication gracefully, and may consequently need to implement mechanisms to detect duplicates. Even if UDP datagram reception triggers only idempotent operations, applications may want to suppress duplicate datagrams to reduce load.

Applications that require ordered delivery MUST reestablish datagram ordering themselves. The Internet can significantly delay some packets with respect to others, e.g., due to routing transients, intermittent connectivity, or mobility. This can cause reordering, where UDP datagrams arrive at the receiver in an order different from the transmission order.

Applications that use multiple transport ports need to be robust to reordering between sessions. Load-balancing techniques within the network, such as Equal Cost Multipath (ECMP) forwarding can also result in a lack of ordering between different transport sessions, even between the same two network endpoints.

It is important to note that the time by which packets are reordered or after which duplicates can still arrive can be very large. Even more importantly, there is no well-defined upper boundary here. [RFC0793] defines the maximum delay a TCP segment should experience -- the Maximum Segment Lifetime (MSL) -- as 2 minutes. No other RFC defines an MSL for other transport protocols or IP itself. The MSL

value defined for TCP is conservative enough that it SHOULD be used by other protocols, including UDP. Therefore, applications SHOULD be robust to the reception of delayed or duplicate packets that are received within this 2-minute interval.

Retransmission of lost packets or messages is a common reliability mechanism. Such retransmissions can increase network load in response to congestion, worsening that congestion. Any application that uses retransmission is responsible for congestion control of its retransmissions (as well as the application's original traffic), and hence is subject to the Congestion Control guidelines in Section 3.1. Guidance on the appropriate measurement of RTT in Section 3.1.1 also applies for timers used for retransmission packet loss detection.

Instead of implementing these relatively complex reliability mechanisms by itself, an application that requires reliable and ordered message delivery SHOULD whenever possible choose an IETF standard transport protocol that provides these features.

3.4. Checksum Guidelines

The UDP header includes an optional, 16-bit one's complement checksum that provides an integrity check. These checks are not strong from a coding or cryptographic perspective, and are not designed to detect physical-layer errors or malicious modification of the datagram [RFC3819]. Application developers SHOULD implement additional checks where data integrity is important, e.g., through a Cyclic Redundancy Check (CRC) or keyed or non-keyed cryptographic hash included with the data to verify the integrity of an entire object/file sent over the UDP service.

The UDP checksum provides a statistical guarantee that the payload was not corrupted in transit. It also allows the receiver to verify that it was the intended destination of the packet, because it covers the IP addresses, port numbers, and protocol number, and it verifies that the packet is not truncated or padded, because it covers the size field. It therefore protects an application against receiving corrupted payload data in place of, or in addition to, the data that was sent. More description of the set of checks performed using the checksum field is provided in Section 3.1 of [RFC6396].

Applications SHOULD enable UDP checksums [RFC1122]. For IPv4, [RFC0768] permits an option to disable their use, by setting a zero checksum value. An application is permitted to optionally discard UDP datagrams with a zero checksum [RFC1122].

When UDP is used over IPv6, the UDP checksum is relied upon to protect both the IPv6 and UDP headers from corruption (because IPv6

lacks a checksum) and MUST be used as specified in [RFC2460]. Under specific conditions a UDP application is allowed to use a zero UDP zero-checksum mode with a tunnel protocol (see Section 3.4.1).

Applications that choose to disable UDP checksums MUST NOT make assumptions regarding the correctness of received data and MUST behave correctly when a UDP datagram is received that was originally sent to a different destination or is otherwise corrupted.

3.4.1. IPv6 Zero UDP Checksum

[RFC6935] defines a method that enables use of a zero UDP zero-checksum mode with a tunnel protocol, providing that the method satisfies the requirements in [RFC6936]. The application MUST implement mechanisms and/or usage restrictions when enabling this mode. This includes defining the scope for usage and measures to prevent leakage of traffic to other UDP applications (see Appendix A Section 3.6). These additional design requirements for using a zero IPv6 UDP checksum are not present for IPv4, since the IPv4 header validates information that is not protected in an IPv6 packet. Key requirements are:

- o Use of the UDP checksum with IPv6 MUST be the default configuration for all implementations [RFC6935]. The receiving endpoint MUST only allow the use of UDP zero-checksum mode for IPv6 on a UDP destination port that is specifically enabled.

- o An application that support a checksum different to that in [RFC2460] MUST comply with all implementation requirements specified in Section 4 of [RFC6936] and with the usage requirements specified in Section 5 of [RFC6936].

- o A UDP application MUST check that the source and destination IPv6 addresses are valid for any packets with a UDP zero-checksum and MUST discard any packet for which this check fails. To protect from misdelivery, new encapsulation designs SHOULD include an integrity check at the transport layer that includes at least the IPv6 header, the UDP header and the shim header for the encapsulation, if any [RFC6936].

- o One way to help satisfy the requirements of [RFC6936] may be to limit the usage of such tunnels, e.g., to constrain traffic to an operator network, as discussed in Section 3.6. The encapsulation defined for MPLS in UDP [RFC7510] chooses this approach.

As in IPv4, IPv6 applications that choose to disable UDP checksums MUST NOT make assumptions regarding the correctness of received data

and MUST behave correctly when a UDP datagram is received that was originally sent to a different destination or is otherwise corrupted.

IPv6 datagrams with a zero UDP checksum will not be passed by any middlebox that validates the checksum based on [RFC2460] or that updates the UDP checksum field, such as NATs or firewalls. Changing this behavior would require such middleboxes to be updated to correctly handle datagrams with zero UDP checksums To ensure end-to-end robustness, applications that may be deployed in the general Internet MUST provide a mechanism to safely fall back to using a checksum when a path change occurs that redirects a zero UDP checksum flow over a path that includes a middlebox that discards IPv6 datagrams with a zero UDP checksum.

3.4.2. UDP-Lite

A special class of applications can derive benefit from having partially-damaged payloads delivered, rather than discarded, when using paths that include error-prone links. Such applications can tolerate payload corruption and MAY choose to use the Lightweight User Datagram Protocol (UDP-Lite) [RFC3828] variant of UDP instead of basic UDP. Applications that choose to use UDP-Lite instead of UDP should still follow the congestion control and other guidelines described for use with UDP in Section 3.

UDP-Lite changes the semantics of the UDP "payload length" field to that of a "checksum coverage length" field. Otherwise, UDP-Lite is semantically identical to UDP. The interface of UDP-Lite differs from that of UDP by the addition of a single (socket) option that communicates the checksum coverage length: at the sender, this specifies the intended checksum coverage, with the remaining unprotected part of the payload called the "error-insensitive part." By default, the UDP-Lite checksum coverage extends across the entire datagram. If required, an application may dynamically modify this length value, e.g., to offer greater protection to some messages. UDP-Lite always verifies that a packet was delivered to the intended destination, i.e., always verifies the header fields. Errors in the insensitive part will not cause a UDP datagram to be discarded by the destination. Applications using UDP-Lite therefore MUST NOT make assumptions regarding the correctness of the data received in the insensitive part of the UDP-Lite payload.

A UDP-Lite sender SHOULD select the minimum checksum coverage to include all sensitive payload information. For example, applications that use the Real-Time Protocol (RTP) [RFC3550] will likely want to protect the RTP header against corruption. Applications, where appropriate, MUST also introduce their own appropriate validity

checks for protocol information carried in the insensitive part of the UDP-Lite payload (e.g., internal CRCs).

A UDP-Lite receiver MUST set a minimum coverage threshold for incoming packets that is not smaller than the smallest coverage used by the sender [RFC3828]. The receiver SHOULD select a threshold that is sufficiently large to block packets with an inappropriately short coverage field. This may be a fixed value, or may be negotiated by an application. UDP-Lite does not provide mechanisms to negotiate the checksum coverage between the sender and receiver. This therefore needs to be performed by the application.

Applications can still experience packet loss when using UDP-Lite. The enhancements offered by UDP-Lite rely upon a link being able to intercept the UDP-Lite header to correctly identify the partial coverage required. When tunnels and/or encryption are used, this can result in UDP-Lite datagrams being treated the same as UDP datagrams, i.e., result in packet loss. Use of IP fragmentation can also prevent special treatment for UDP-Lite datagrams, and this is another reason why applications SHOULD avoid IP fragmentation (Section 3.2).

UDP-Lite is supported in some endpoint protocol stacks. Current support for middlebox traversal using UDP-Lite is poor, because UDP-Lite uses a different IPv4 protocol number or IPv6 "next header" value than that used for UDP; therefore, few middleboxes are currently able to interpret UDP-Lite and take appropriate actions when forwarding the packet. This makes UDP-Lite less suited for applications needing general Internet support, until such time as UDP-Lite has achieved better support in middleboxes.

3.5. Middlebox Traversal Guidelines

Network address translators (NATs) and firewalls are examples of intermediary devices ("middleboxes") that can exist along an end-to-end path. A middlebox typically performs a function that requires it to maintain per-flow state. For connection-oriented protocols, such as TCP, middleboxes snoop and parse the connection-management information, and create and destroy per-flow state accordingly. For a connectionless protocol such as UDP, this approach is not possible. Consequently, middleboxes can create per-flow state when they see a packet that -- according to some local criteria -- indicates a new flow, and destroy the state after some time during which no packets belonging to the same flow have arrived.

Depending on the specific function that the middlebox performs, this behavior can introduce a time-dependency that restricts the kinds of UDP traffic exchanges that will be successful across the middlebox. For example, NATs and firewalls typically define the partial path on

one side of them to be interior to the domain they serve, whereas the partial path on their other side is defined to be exterior to that domain. Per-flow state is typically created when the first packet crosses from the interior to the exterior, and while the state is present, NATs and firewalls will forward return traffic. Return traffic that arrives after the per-flow state has timed out is dropped, as is other traffic that arrives from the exterior.

Many applications that use UDP for communication operate across middleboxes without needing to employ additional mechanisms. One example is the Domain Name System (DNS), which has a strict request-response communication pattern that typically completes within seconds.

Other applications may experience communication failures when middleboxes destroy the per-flow state associated with an application session during periods when the application does not exchange any UDP traffic. Applications SHOULD be able to gracefully handle such communication failures and implement mechanisms to re-establish application-layer sessions and state.

For some applications, such as media transmissions, this re-synchronization is highly undesirable, because it can cause user-perceivable playback artifacts. Such specialized applications MAY send periodic keep-alive messages to attempt to refresh middlebox state (e.g., [RFC7675]). It is important to note that keep-alive messages are not recommended for general use -- they are unnecessary for many applications and can consume significant amounts of system and network resources.

An application that needs to employ keep-alive messages to deliver useful service over UDP in the presence of middleboxes SHOULD NOT transmit them more frequently than once every 15 seconds and SHOULD use longer intervals when possible. No common timeout has been specified for per-flow UDP state for arbitrary middleboxes. NATs require a state timeout of 2 minutes or longer [RFC4787]. However, empirical evidence suggests that a significant fraction of currently deployed middleboxes unfortunately use shorter timeouts. The timeout of 15 seconds originates with the Interactive Connectivity Establishment (ICE) protocol [RFC5245]. When an application is deployed in a controlled environment, the deployer SHOULD investigate whether the target environment allows applications to use longer intervals, or whether it offers mechanisms to explicitly control middlebox state timeout durations, for example, using the Port Control Protocol (PCP) [RFC6887], Middlebox Communications (MIDCOM) [RFC3303], Next Steps in Signaling (NSIS) [RFC5973], or Universal Plug and Play (UPnP) [UPnP]. It is RECOMMENDED that applications apply slight random variations ("jitter") to the timing of keep-alive

transmissions, to reduce the potential for persistent synchronization between keep-alive transmissions from different hosts [RFC7675].

Sending keep-alive messages is not a substitute for implementing a mechanism to recover from broken sessions. Like all UDP datagrams, keep-alive messages can be delayed or dropped, causing middlebox state to time out. In addition, the congestion control guidelines in Section 3.1 cover all UDP transmissions by an application, including the transmission of middlebox keep-alive messages. Congestion control may thus lead to delays or temporary suspension of keep-alive transmission.

Keep-alive messages are NOT RECOMMENDED for general use. They are unnecessary for many applications and may consume significant resources. For example, on battery-powered devices, if an application needs to maintain connectivity for long periods with little traffic, the frequency at which keep-alive messages are sent can become the determining factor that governs power consumption, depending on the underlying network technology.

Because many middleboxes are designed to require keep-alive messages for TCP connections at a frequency that is much lower than that needed for UDP, this difference alone can often be sufficient to prefer TCP over UDP for these deployments. On the other hand, there is anecdotal evidence that suggests that direct communication through middleboxes, e.g., by using ICE [RFC5245], does succeed less often with TCP than with UDP. The trade-offs between different transport protocols -- especially when it comes to middlebox traversal -- deserve careful analysis.

UDP applications that could be deployed in the Internet need to be designed understanding that there are many variants of middlebox behavior, and although UDP is connectionless, middleboxes often maintain state for each UDP flow. Using multiple UDP flows can consume available state space and also can lead to changes in the way the middlebox handles subsequent packets (either to protect its internal resources, or to prevent perceived misuse). The probability of path failure can increase when applications use multiple UDP flows in parallel (see Section 5.1.2 for recommendations on usage of multiple ports).

3.6. Limited Applicability and Controlled Environments

Two different types of applicability have been identified for the specification of IETF applications that utilize UDP:

General Internet. By default, IETF specifications target deployment on the general Internet. Experience has shown that successful

protocols developed in one specific context or for a particular application tend to become used in a wider range of contexts. For example, a protocol with an initial deployment within a local area network may subsequently be used over a virtual network that traverses the Internet, or in the Internet in general. Applications designed for general Internet use may experience a range of network device behaviors, and in particular should consider whether applications need to operate over paths that may include middleboxes.

Controlled Environment. A protocol/encapsulation/tunnel could be designed to be used only within a controlled environment. For example, an application designed for use by a network operator might only be deployed within the network of that single network operator or on networks of an adjacent set of cooperating network operators. The application traffic may then be managed to avoid congestion, rather than relying on built-in mechanisms, which are required when operating over the general Internet. Applications that target a limited applicability use case may be able to take advantage of specific hardware (e.g., carrier-grade equipment) or underlying protocol features of the subnetwork over which they are used.

Specifications addressing a limited applicability use case or a controlled environment SHOULD identify how in their restricted deployment a level of safety is provided that is equivalent to that of a protocol designed for operation over the general Internet (e.g., a design based on extensive experience with deployments of particular methods that provide features that cannot be expected in general Internet equipment and the robustness of the design of MPLS to corruption of headers both helped justify use of an alternate UDP integrity check [RFC7510]).

An IETF specification targeting a controlled environment is expected to provide an applicability statement that restricts the application traffic to the controlled environment, and would be expected to describe how methods can be provided to discourage or prevent escape of corrupted packets from the environment (for example, section 5 of [RFC7510]).

4. Multicast UDP Usage Guidelines

This section complements Section 3 by providing additional guidelines that are applicable to multicast and broadcast usage of UDP.

Multicast and broadcast transmission [RFC1112] usually employ the UDP transport protocol, although they may be used with other transport protocols (e.g., UDP-Lite).

There are currently two models of multicast delivery: the Any-Source Multicast (ASM) model as defined in [RFC1112] and the Source-Specific Multicast (SSM) model as defined in [RFC4607]. ASM group members will receive all data sent to the group by any source, while SSM constrains the distribution tree to only one single source.

Specialized classes of applications also use UDP for IP multicast or broadcast [RFC0919]. The design of such specialized applications requires expertise that goes beyond simple, unicast-specific guidelines, since these senders may transmit to potentially very many receivers across potentially very heterogeneous paths at the same time, which significantly complicates congestion control, flow control, and reliability mechanisms.

This section provides guidance on multicast and broadcast UDP usage. Use of broadcast by an application is normally constrained by routers to the local subnetwork. However, use of tunneling techniques and proxies can and does result in some broadcast traffic traversing Internet paths. These guidelines therefore also apply to broadcast traffic.

The IETF has defined a reliable multicast framework [RFC3048] and several building blocks to aid the designers of multicast applications, such as [RFC3738] or [RFC4654].

Senders to anycast destinations must be aware that successive messages sent to the same anycast IP address may be delivered to different anycast nodes, i.e., arrive at different locations in the topology.

Most UDP tunnels that carry IP multicast traffic use a tunnel encapsulation with a unicast destination address, such as Automatic Multicast Tunneling [RFC7450]. These MUST follow the same requirements as a tunnel carrying unicast data (see Section 3.1.11). There are deployment cases and solutions where the outer header of a UDP tunnel contains a multicast destination address, such as [RFC6513]. These cases are primarily deployed in controlled environments over reserved capacity, often operating within a single administrative domain, or between two domains over a bi-laterally agreed upon path with reserved capacity, and so congestion control is OPTIONAL, but circuit breaker techniques are still RECOMMENDED in order to restore some degree of service should the offered load exceed the reserved capacity (e.g., due to misconfiguration).

4.1. Multicast Congestion Control Guidelines

Unicast congestion-controlled transport mechanisms are often not applicable to multicast distribution services, or simply do not scale to large multicast trees, since they require bi-directional communication and adapt the sending rate to accommodate the network conditions to a single receiver. In contrast, multicast distribution trees may fan out to massive numbers of receivers, which limits the scalability of an in-band return channel to control the sending rate, and the one-to-many nature of multicast distribution trees prevents adapting the rate to the requirements of an individual receiver. For this reason, generating TCP-compatible aggregate flow rates for Internet multicast data, either native or tunneled, is the responsibility of the application implementing the congestion control.

Applications using multicast SHOULD provide appropriate congestion control. Multicast congestion control needs to be designed using mechanisms that are robust to the potential heterogeneity of both the multicast distribution tree and the receivers belonging to a group. Heterogeneity may manifest itself in some receivers experiencing more loss that others, higher delay, and/or less ability to respond to network conditions. Congestion control is particularly important for any multicast session where all or part of the multicast distribution tree spans an access network (e.g., a home gateway). Two styles of congestion control have been defined in the RFC-series:

- o Feedback-based congestion control, in which the sender receives multicast or unicast UDP messages from the receivers allowing it to assess the level of congestion and then adjust the sender rate(s) (e.g., [RFC5740],[RFC4654]). Multicast methods may operate on longer timescales than for unicast (e.g., due to the higher group RTT of a heterogeneous group). A control method could decide not to reduce the rate of the entire multicast group in response to a control message received from a single receiver (e.g., a sender could set a minimum rate and decide to request a congested receiver to leave the multicast group and could also decide to distribute content to these congested receivers at a lower rate using unicast congestion control).

- o Receiver-driven congestion control, which does not require a receiver to send explicit UDP control messages for congestion control (e.g., [RFC3738], [RFC5775]). Instead, the sender distributes the data across multiple IP multicast groups (e.g., using a set of {S,G} channels). Each receiver determines its own level of congestion and controls its reception rate using only multicast join/leave messages sent in the network control plane. This method scales to arbitrary large groups of receivers.

Any multicast-enabled receiver may attempt to join and receive traffic from any group. This may imply the need for rate limits on individual receivers or the aggregate multicast service. Note there is no way at the transport layer to prevent a join message propagating to the next-hop router.

Some classes of multicast applications support applications that can monitor the user-level quality of the transfer at the receiver. Applications that can detect a significant reduction in user quality SHOULD regard this as a congestion signal (e.g., to leave a group using layered multicast encoding) or, if not, SHOULD use this signal to provide a circuit breaker to terminate the flow by leaving the multicast group.

4.1.1. Bulk Transfer Multicast Applications

Applications that perform bulk transmission of data over a multicast distribution tree, i.e., applications that exchange more than a few UDP datagrams per RTT, SHOULD implement a method for congestion control. The currently RECOMMENDED IETF methods are: Asynchronous Layered Coding (ALC) [RFC5775], TCP-Friendly Multicast Congestion Control (TFMCC) [RFC4654], Wave and Equation Based Rate Control (WEBRC) [RFC3738], NACK-Oriented Reliable Multicast (NORM) transport protocol [RFC5740], File Delivery over Unidirectional Transport (FLUTE) [RFC6726], Real Time Protocol/Control Protocol (RTP/RTCP) [RFC3550].

An application can alternatively implement another congestion control schemes following the guidelines of [RFC2887] and utilizing the framework of [RFC3048]. Bulk transfer applications that choose not to implement [RFC4654], [RFC5775], [RFC3738], [RFC5740], [RFC6726], or [RFC3550] SHOULD implement a congestion control scheme that results in bandwidth use that competes fairly with TCP within an order of magnitude.

Section 2 of [RFC3551] states that multimedia applications SHOULD monitor the packet loss rate to ensure that it is within acceptable parameters. Packet loss is considered acceptable if a TCP flow across the same network path under the same network conditions would achieve an average throughput, measured on a reasonable timescale, that is not less than that of the UDP flow. The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in timescale and throughput.

4.1.2. Low Data-Volume Multicast Applications

All the recommendations in Section 3.1.3 are also applicable to low data-volume multicast applications.

4.2. Message Size Guidelines for Multicast

A multicast application SHOULD NOT send UDP datagrams that result in IP packets that exceed the effective MTU as described in section 3 of [RFC6807]. Consequently, an application SHOULD either use the effective MTU information provided by the Population Count Extensions to Protocol Independent Multicast [RFC6807] or implement path MTU discovery itself (see Section 3.2) to determine whether the path to each destination will support its desired message size without fragmentation.

5. Programming Guidelines

The de facto standard application programming interface (API) for TCP/IP applications is the "sockets" interface [POSIX]. Some platforms also offer applications the ability to directly assemble and transmit IP packets through "raw sockets" or similar facilities. This is a second, more cumbersome method of using UDP. The guidelines in this document cover all such methods through which an application may use UDP. Because the sockets API is by far the most common method, the remainder of this section discusses it in more detail.

Although the sockets API was developed for UNIX in the early 1980s, a wide variety of non-UNIX operating systems also implement it. The sockets API supports both IPv4 and IPv6 [RFC3493]. The UDP sockets API differs from that for TCP in several key ways. Because application programmers are typically more familiar with the TCP sockets API, this section discusses these differences. [STEVENS] provides usage examples of the UDP sockets API.

UDP datagrams may be directly sent and received, without any connection setup. Using the sockets API, applications can receive packets from more than one IP source address on a single UDP socket. Some servers use this to exchange data with more than one remote host through a single UDP socket at the same time. Many applications need to ensure that they receive packets from a particular source address; these applications MUST implement corresponding checks at the application layer or explicitly request that the operating system filter the received packets.

Many operating systems also allow a UDP socket to be connected, i.e., to bind a UDP socket to a specific pair of addresses and ports. This

is similar to the corresponding TCP sockets API functionality. However, for UDP, this is only a local operation that serves to simplify the local send/receive functions and to filter the traffic for the specified addresses and ports. Binding a UDP socket does not establish a connection -- UDP does not notify the remote end when a local UDP socket is bound. Binding a socket also allows configuring options that affect the UDP or IP layers, for example, use of the UDP checksum or the IP Timestamp option. On some stacks, a bound socket also allows an application to be notified when ICMP error messages are received for its transmissions [RFC1122].

If a client/server application executes on a host with more than one IP interface, the application SHOULD send any UDP responses with an IP source address that matches the IP destination address of the UDP datagram that carried the request (see [RFC1122], Section 4.1.3.5). Many middleboxes expect this transmission behavior and drop replies that are sent from a different IP address, as explained in Section 3.5.

A UDP receiver can receive a valid UDP datagram with a zero-length payload. Note that this is different from a return value of zero from a read() socket call, which for TCP indicates the end of the connection.

UDP provides no flow-control, i.e., the sender at any given time does not know whether the receiver is able to handle incoming transmissions. This is another reason why UDP-based applications need to be robust in the presence of packet loss. This loss can also occur within the sending host, when an application sends data faster than the line rate of the outbound network interface. It can also occur at the destination, where receive calls fail to return all the data that was sent when the application issues them too infrequently (i.e., such that the receive buffer overflows). Robust flow control mechanisms are difficult to implement, which is why applications that need this functionality SHOULD consider using a full-featured transport protocol such as TCP.

When an application closes a TCP, SCTP or DCCP socket, the transport protocol on the receiving host is required to maintain TIME-WAIT state. This prevents delayed packets from the closed connection instance from being mistakenly associated with a later connection instance that happens to reuse the same IP address and port pairs. The UDP protocol does not implement such a mechanism. Therefore, UDP-based applications need to be robust to reordering and delay. One application may close a socket or terminate, followed in time by another application receiving on the same port. This later application may then receive packets intended for the first application that were delayed in the network.

5.1. Using UDP Ports

The rules and procedures for the management of the Service Name and Transport Protocol Port Number Registry are specified in [RFC6335]. Recommendations for use of UDP ports are provided in [RFC7605].

A UDP sender SHOULD NOT use a source port value of zero. A source port number that cannot be easily determined from the address or payload type provides protection at the receiver from data injection attacks by off-path devices. A UDP receiver SHOULD NOT bind to port zero.

Applications SHOULD implement receiver port and address checks at the application layer or explicitly request that the operating system filter the received packets to prevent receiving packets with an arbitrary port. This measure is designed to provide additional protection from data injection attacks from an off-path source (where the port values may not be known).

Applications SHOULD provide a check that protects from off-path data injection, avoiding an application receiving packets that were created by an unauthorized third party. TCP stacks commonly use a randomized source port to provide this protection [RFC6056]; UDP applications should follow the same technique. Middleboxes and end systems often make assumptions about the system ports or user ports, hence it is recommended to use randomized ports in the Dynamic and/or Private Port range. Setting a "randomized" source port also provides greater assurance that reported ICMP errors originate from network systems on the path used by a particular flow. Some UDP applications choose to use a predetermined value for the source port (including some multicast applications), these applications need to therefore employ a different technique. Protection from off-path data attacks can also be provided by randomizing the initial value of another protocol field within the datagram payload, and checking the validity of this field at the receiver (e.g., RTP has random initial sequence number and random media timestamp offsets [RFC3550]).

When using multicast, IP routers perform a reverse-path forwarding (RPF) check for each multicast packet. This provides protection from off-path data injection. When a receiver joins a multicast group and filters based on the source address the filter verifies the sender’s IP address. This is always the case when using a SSM {S,G} channel.

5.1.1. Usage of UDP for source port entropy and the IPv6 Flow Label

Some applications use the UDP datagram header as a source of entropy for network devices that implement ECMP [RFC6438]. A UDP tunnel application targeting this usage, encapsulates an inner packet using

UDP, where the UDP source port value forms a part of the entropy that can be used to balance forwarding of network traffic by the devices that use ECMP. A sending tunnel endpoint selects a source port value in the UDP datagram header that is computed from the inner flow information (e.g., the encapsulated packet headers). To provide sufficient entropy the sending tunnel endpoint maps the encapsulated traffic to one of a range of UDP source values. The value SHOULD be within the ephemeral port range, i.e., 49152 to 65535, where the high order two bits of the port are set to one. The available source port entropy of 14 bits (using the ephemeral port range) plus the outer IP addresses seems sufficient for entropy for most ECMP applications [I-D.ietf-rtgwg-dt-encap].

To avoid reordering within an IP flow, the same UDP source port value SHOULD be used for all packets assigned to an encapsulated flow (e.g., using a hash of the relevant headers). The entropy mapping for a flow MAY change over the lifetime of the encapsulated flow [I-D.ietf-rtgwg-dt-encap]. For instance, this could be changed as a Denial of Service (DOS) mitigation, or as a means to effect routing through the ECMP network. However, the source port selected for a flow SHOULD NOT change more than once in every thirty seconds (e.g., as in [I-D.ietf-tsvwg-gre-in-udp-encap]).

The use of the source port field for entropy has several side-effects that need to be considered, including:

- It can increase the probability of misdelivery of corrupted packets, which increases the need for checksum computation or an equivalent mechanism to protect other UDP applications from misdelivery errors Section 3.4.

- It is expected to reduce the probability of successful middlebox traversal Section 3.5. This use of the source port field will often not be suitable for applications targeting deployment in the general Internet.

- It can prevent the field being usable to protect from off-path attacks (described in Section 5.1). Designers therefore need to consider other mechanisms to provide equivalent protection (e.g., to restrict use to a controlled environment [RFC7510] Section 3.6).

The UDP source port number field has also been leveraged to produce entropy with IPv6. However, in the case of IPv6, the "flow label" [RFC6437] may also alternatively be used as entropy for load balancing [RFC6438]. This use of the flow label for load balancing is consistent with the definition of the field, although further clarity was needed to ensure the field can be consistently used for

this purpose. Therefore, an updated IPv6 flow label [RFC6437] and ECMP routing [RFC6438] usage was specified.

To ensure future opportunities to use the flow label, UDP applications SHOULD set the flow label field, even when an entropy value is also set in the source port field (e.g., An IPv6 tunnel endpoint could copy the source port flow entropy value to the IPv6 flow label field [I-D.ietf-tsvwg-gre-in-udp-encap]). Router vendors are encouraged to start using the IPv6 flow label as a part of the flow hash, providing support for IP-level ECMP without requiring use of UDP. The end-to-end use of flow labels for load balancing is a long-term solution. Even if the usage of the flow label has been clarified, there will be a transition time before a significant proportion of endpoints start to assign a good quality flow label to the flows that they originate. The use of load balancing using the transport header fields will likely continue until widespread deployment is finally achieved.

5.1.2. Applications using Multiple UDP Ports

A single application may exchange several types of data. In some cases, this may require multiple UDP flows (e.g., multiple sets of flows, identified by different five-tuples). [RFC6335] recommends application developers not to apply to IANA to be assigned multiple well-known ports (user or system). This does not discuss the implications of using multiple flows with the same well-known port or pairs of dynamic ports (e.g., identified by a service name or signaling protocol).

Use of multiple flows can affect the network in several ways:

- Starting a series of successive connections can increase the number of state bindings in middleboxes (e.g., NAPT or Firewall) along the network path. UDP-based middlebox traversal usually relies on timeouts to remove old state, since middleboxes are unaware when a particular flow ceases to be used by an application.

- Using several flows at the same time may result in seeing different network characteristics for each flow. It cannot be assumed both follow the same path (e.g., when ECMP is used, traffic is intentionally hashed onto different parallel paths based on the port numbers).

- Using several flows can also increase the occupancy of a binding or lookup table in a middlebox (e.g., NAPT or Firewall), which may cause the device to change the way it manages the flow state.

- o Further, using excessive numbers of flows can degrade the ability of a unicast congestion control to react to congestion events, unless the congestion state is shared between all flows in a session. A receiver-driven multicast congestion control requires the sending application to distribute its data over a set of IP multicast groups, each receiver is therefore expected to receive data from a modest number of simultaneously active UDP ports.

Therefore, applications MUST NOT assume consistent behavior of middleboxes when multiple UDP flows are used; many devices respond differently as the number of used ports increases. Using multiple flows with different QoS requirements requires applications to verify that the expected performance is achieved using each individual flow (five-tuple), see Section 3.1.9.

5.2. ICMP Guidelines

Applications can utilize information about ICMP error messages that the UDP layer passes up for a variety of purposes [RFC1122]. Applications SHOULD appropriately validate the payload of ICMP messages to ensure these are received in response to transmitted traffic (i.e., a reported error condition that corresponds to a UDP datagram actually sent by the application). This requires context, such as local state about communication instances to each destination, that although readily available in connection-oriented transport protocols is not always maintained by UDP-based applications. Note that not all platforms have the necessary APIs to support this validation, and some platforms already perform this validation internally before passing ICMP information to the application.

Any application response to ICMP error messages SHOULD be robust to temporary routing failures (sometimes called "soft errors"), e.g., transient ICMP "unreachable" messages ought to not normally cause a communication abort.

ICMP messages are being increasingly filtered by middleboxes. A UDP application therefore SHOULD NOT rely on their delivery for correct and safe operation.

6. Security Considerations

UDP does not provide communications security. Applications that need to protect their communications against eavesdropping, tampering, or message forgery SHOULD employ end-to-end security services provided by other IETF protocols.

UDP applications SHOULD provide protection from off-path data injection attacks using a randomized source port or equivalent technique (see Section 5.1).

Applications that respond to short requests with potentially large responses are a potential vector for amplification attacks, and SHOULD take steps to minimize their potential for being abused as part of a DoS attack. That could mean authenticating the sender before responding; noting that the source IP address of a request is not a useful authenticator, because it can easily be spoofed. Or it may mean otherwise limiting the cases where short unauthenticated requests produce large responses. Applications MAY also want to offer ways to limit the number of requests they respond to in a time interval, in order to cap the bandwidth they consume.

One option for securing UDP communications is with IPsec [RFC4301], which can provide authentication for flows of IP packets through the Authentication Header (AH) [RFC4302] and encryption and/or authentication through the Encapsulating Security Payload (ESP) [RFC4303]. Applications use the Internet Key Exchange (IKE) [RFC7296] to configure IPsec for their sessions. Depending on how IPsec is configured for a flow, it can authenticate or encrypt the UDP headers as well as UDP payloads. If an application only requires authentication, ESP with no encryption but with authentication is often a better option than AH, because ESP can operate across middleboxes. An application that uses IPsec requires the support of an operating system that implements the IPsec protocol suite, and the network path must permit IKE and IPsec traffic. This may become more common with IPv6 deployments [RFC6092].

Although it is possible to use IPsec to secure UDP communications, not all operating systems support IPsec or allow applications to easily configure it for their flows. A second option for securing UDP communications is through Datagram Transport Layer Security (DTLS) [RFC6347][RFC7525]. DTLS provides communication privacy by encrypting UDP payloads. It does not protect the UDP headers. Applications can implement DTLS without relying on support from the operating system.

Many other options for authenticating or encrypting UDP payloads exist. For example, the GSS-API security framework [RFC2743] or Cryptographic Message Syntax (CMS) [RFC5652] could be used to protect UDP payloads. There exist a number of security options for RTP [RFC3550] over UDP, especially to accomplish key-management, see [RFC7201]. These options covers many usages, including point-to-point, centralized group communication as well as multicast. In some applications, a better solution is to protect larger stand-alone objects, such as files or messages, instead of individual UDP

payloads. In these situations, CMS [RFC5652], S/MIME [RFC5751] or OpenPGP [RFC4880] could be used. In addition, there are many non-IETF protocols in this area.

Like congestion control mechanisms, security mechanisms are difficult to design and implement correctly. It is hence RECOMMENDED that applications employ well-known standard security mechanisms such as DTLS or IPsec, rather than inventing their own.

The Generalized TTL Security Mechanism (GTSM) [RFC5082] may be used with UDP applications when the intended endpoint is on the same link as the sender. This lightweight mechanism allows a receiver to filter unwanted packets.

In terms of congestion control, [RFC2309] and [RFC2914] discuss the dangers of congestion-unresponsive flows to the Internet. [I-D.ietf-tsvwg-circuit-breaker] describes methods that can be used to set a performance envelope that can assist in preventing congestion collapse in the absence of congestion control or when the congestion control fails to react to congestion events. This document provides guidelines to designers of UDP-based applications to congestion-control their transmissions, and does not raise any additional security concerns.

Some network operators have experienced surges of UDP attack traffic that are multiple orders of magnitude above the baseline traffic rate for UDP. This can motivate operators to limit the data rate or packet rate of UDP traffic. This may in turn limit the throughput that an application can achieve using UDP and could also result in higher packet loss for UDP traffic that would not be experienced if other transport protocols had been used.

A UDP application with a long-lived association between the sender and receiver, ought to be designed so that the sender periodically checks that the receiver still wants ("consents") to receive traffic and need to be designed to stop if there is no explicit confirmation of this [RFC7675]. Applications that require communications in two directions to implement protocol functions (such as reliability or congestion control) will need to independently check both directions of communication, and may have to exchange keep-alive messages to traverse middleboxes (see Section 3.5).

7. Summary

This section summarizes the key guidelines made in Sections 3 - 6 in a tabular format (Table 1) for easy referencing.

```
+-------------------------------------------------------------+---------+
```

Recommendation	Section
MUST tolerate a wide range of Internet path conditions	3
SHOULD use a full-featured transport (e.g., TCP)	
SHOULD control rate of transmission	3.1
SHOULD perform congestion control over all traffic	
for bulk transfers,	3.1.2
SHOULD consider implementing TFRC	
else, SHOULD in other ways use bandwidth similar to TCP	
for non-bulk transfers,	3.1.3
SHOULD measure RTT and transmit max. 1 datagram/RTT	3.1.1
else, SHOULD send at most 1 datagram every 3 seconds	
SHOULD back-off retransmission timers following loss	
SHOULD provide mechanisms to regulate the bursts of	3.1.6
transmission	
MAY implement ECN; a specific set of application	3.1.7
mechanisms are REQUIRED if ECN is used.	
for DiffServ, SHOULD NOT rely on implementation of PHBs	3.1.8
for QoS-enabled paths, MAY choose not to use CC	3.1.9
SHOULD NOT rely solely on QoS for their capacity	3.1.10
non-CC controlled flows SHOULD implement a transport	
circuit breaker	
MAY implement a circuit breaker for other applications	
for tunnels carrying IP traffic,	3.1.11
SHOULD NOT perform congestion control	
MUST correctly process the IP ECN field	
for non-IP tunnels or rate not determined by traffic,	
SHOULD perform CC or use circuit breaker	3.1.11
SHOULD restrict types of traffic transported by the	
tunnel	
SHOULD NOT send datagrams that exceed the PMTU, i.e.,	3.2
SHOULD discover PMTU or send datagrams < minimum PMTU;	
Specific application mechanisms are REQUIRED if PLPMTUD	
is used.	
SHOULD handle datagram loss, duplication, reordering	3.3
SHOULD be robust to delivery delays up to 2 minutes	

SHOULD enable IPv4 UDP checksum	3.4
SHOULD enable IPv6 UDP checksum; Specific application mechanisms are REQUIRED if a zero IPv6 UDP checksum is used.	3.4.1
SHOULD provide protection from off-path attacks	5.1
else, MAY use UDP-Lite with suitable checksum coverage	3.4.2
SHOULD NOT always send middlebox keep-alive messages MAY use keep-alives when needed (min. interval 15 sec)	3.5
Applications specified for use in limited use (or controlled environments) SHOULD identify equivalent mechanisms and describe their use-case.	3.6
Bulk multicast apps SHOULD implement congestion control	4.1.1
Low volume multicast apps SHOULD implement congestion control	4.1.2
Multicast apps SHOULD use a safe PMTU	4.2
SHOULD avoid using multiple ports MUST check received IP source address	5.1.2
SHOULD validate payload in ICMP messages	5.2
SHOULD use a randomized source port or equivalent technique, and, for client/server applications, SHOULD send responses from source address matching request	6
SHOULD use standard IETF security protocols when needed	6

Table 1: Summary of recommendations

8. IANA Considerations

Note to RFC-Editor: please remove this entire section prior to publication.

This document raises no IANA considerations.

9. Acknowledgments

The middlebox traversal guidelines in Section 3.5 incorporate ideas from Section 5 of [I-D.ford-behave-app] by Bryan Ford, Pyda Srisuresh, and Dan Kegel. The protocol timer guidelines in Section 3.1.1 were largely contributed by Mark Allman.

G. Fairhurst received funding from the European Union’s Horizon 2020 research and innovation program 2014-2018 under grant agreement No. 644334 (NEAT). Lars Eggert has received funding from the European Union’s Horizon 2020 research and innovation program 2014-2018 under grant agreement No. 644866 (SSICLOPS). This document reflects only the authors’ views and the European Commission is not responsible for any use that may be made of the information it contains.

10. References

10.1. Normative References

[I-D.ietf-tsvwg-circuit-breaker]
Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-15 (work in progress), April 2016.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <http://www.rfc-editor.org/info/rfc1191>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <http://www.rfc-editor.org/info/rfc2914>.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., Ed., and G. Fairhurst, Ed., "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, DOI 10.17487/RFC3828, July 2004, <http://www.rfc-editor.org/info/rfc3828>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <http://www.rfc-editor.org/info/rfc4787>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <http://www.rfc-editor.org/info/rfc4821>.

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <http://www.rfc-editor.org/info/rfc5348>.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <http://www.rfc-editor.org/info/rfc6040>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP’s Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <http://www.rfc-editor.org/info/rfc6298>.

10.2. Informative References

[ALLMAN] Allman, M. and E. Blanton, "Notes on burst mitigation for transport protocols", March 2005.

[FABER] Faber, T., Touch, J., and W. Yue, "The TIME-WAIT State in TCP and Its Effect on Busy Servers", Proc. IEEE Infocom, March 1999.

[I-D.ford-behave-app] Ford, B., "Application Design Guidelines for Traversal through Network Address Translators", draft-ford-behave-app-05 (work in progress), March 2007.

[I-D.ietf-aqm-ecn-benefits] Fairhurst, G. and M. Welzl, "The Benefits of using Explicit Congestion Notification (ECN)", draft-ietf-aqm-ecn-benefits-08 (work in progress), November 2015.

[I-D.ietf-avtcore-rtp-circuit-breakers] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", draft-ietf-avtcore-rtp-circuit-breakers-18 (work in progress), August 2016.

[I-D.ietf-intarea-tunnels] Touch, J. and W. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-03 (work in progress), July 2016.

[I-D.ietf-rtgwg-dt-encap] Nordmark, E., Tian, A., Gross, J., Hudson, J., Kreeger, L., Garg, P., Thaler, P., and T. Herbert, "Encapsulation Considerations", draft-ietf-rtgwg-dt-encap-01 (work in progress), March 2016.

[I-D.ietf-tsvwg-gre-in-udp-encap] Yong, L., Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", draft-ietf-tsvwg-gre-in-udp-encap-19 (work in progress), September 2016.

[POSIX] IEEE Std. 1003.1-2001, , "Standard for Information Technology - Portable Operating System Interface (POSIX)", Open Group Technical Standard: Base Specifications Issue 6, ISO/IEC 9945:2002, December 2001.

[RFC0919] Mogul, J., "Broadcasting Internet Datagrams", STD 5, RFC 919, DOI 10.17487/RFC0919, October 1984, <http://www.rfc-editor.org/info/rfc919>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <http://www.rfc-editor.org/info/rfc1112>.

[RFC1536] Kumar, A., Postel, J., Neuman, C., Danzig, P., and S. Miller, "Common DNS Implementation Errors and Suggested Fixes", RFC 1536, DOI 10.17487/RFC1536, October 1993, <http://www.rfc-editor.org/info/rfc1536>.

[RFC1546] Partridge, C., Mendez, T., and W. Milliken, "Host Anycasting Service", RFC 1546, DOI 10.17487/RFC1546, November 1993, <http://www.rfc-editor.org/info/rfc1546>.

[RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, DOI 10.17487/RFC2309, April 1998, <http://www.rfc-editor.org/info/rfc2309>.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <http://www.rfc-editor.org/info/rfc2475>.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <http://www.rfc-editor.org/info/rfc2675>.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <http://www.rfc-editor.org/info/rfc2743>.

[RFC2887] Handley, M., Floyd, S., Whetten, B., Kermode, R., Vicisano, L., and M. Luby, "The Reliable Multicast Design Space for Bulk Data Transfer", RFC 2887, DOI 10.17487/RFC2887, August 2000, <http://www.rfc-editor.org/info/rfc2887>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <http://www.rfc-editor.org/info/rfc2983>.

[RFC3048] Whetten, B., Vicisano, L., Kermode, R., Handley, M., Floyd, S., and M. Luby, "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", RFC 3048, DOI 10.17487/RFC3048, January 2001, <http://www.rfc-editor.org/info/rfc3048>.

[RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, DOI 10.17487/RFC3124, June 2001, <http://www.rfc-editor.org/info/rfc3124>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.

[RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, DOI 10.17487/RFC3303, August 2002, <http://www.rfc-editor.org/info/rfc3303>.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <http://www.rfc-editor.org/info/rfc3493>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <http://www.rfc-editor.org/info/rfc3551>.

[RFC3738] Luby, M. and V. Goyal, "Wave and Equation Based Rate Control (WEBRC) Building Block", RFC 3738, DOI 10.17487/RFC3738, April 2004, <http://www.rfc-editor.org/info/rfc3738>.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <http://www.rfc-editor.org/info/rfc3758>.

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <http://www.rfc-editor.org/info/rfc3819>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <http://www.rfc-editor.org/info/rfc4301>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <http://www.rfc-editor.org/info/rfc4302>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <http://www.rfc-editor.org/info/rfc4340>.

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, DOI 10.17487/RFC4341, March 2006, <http://www.rfc-editor.org/info/rfc4341>.

[RFC4342] Floyd, S., Kohler, E., and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, DOI 10.17487/RFC4342, March 2006, <http://www.rfc-editor.org/info/rfc4342>.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <http://www.rfc-editor.org/info/rfc4380>.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <http://www.rfc-editor.org/info/rfc4607>.

[RFC4654] Widmer, J. and M. Handley, "TCP-Friendly Multicast Congestion Control (TFMCC): Protocol Specification", RFC 4654, DOI 10.17487/RFC4654, August 2006, <http://www.rfc-editor.org/info/rfc4654>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <http://www.rfc-editor.org/info/rfc4880>.

[RFC4890] Davies, E. and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls", RFC 4890, DOI 10.17487/RFC4890, May 2007, <http://www.rfc-editor.org/info/rfc4890>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <http://www.rfc-editor.org/info/rfc4963>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <http://www.rfc-editor.org/info/rfc4987>.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, DOI 10.17487/RFC5082, October 2007, <http://www.rfc-editor.org/info/rfc5082>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <http://www.rfc-editor.org/info/rfc5245>.

[RFC5622] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion ID 4: TCP-Friendly Rate Control for Small Packets (TFRC-SP)", RFC 5622, DOI 10.17487/RFC5622, August 2009, <http://www.rfc-editor.org/info/rfc5622>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, DOI 10.17487/RFC5740, November 2009, <http://www.rfc-editor.org/info/rfc5740>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <http://www.rfc-editor.org/info/rfc5751>.

[RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, DOI 10.17487/RFC5775, April 2010, <http://www.rfc-editor.org/info/rfc5775>.

[RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", RFC 5971, DOI 10.17487/RFC5971, October 2010, <http://www.rfc-editor.org/info/rfc5971>.

[RFC5973] Stiemerling, M., Tschofenig, H., Aoun, C., and E. Davies, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", RFC 5973, DOI 10.17487/RFC5973, October 2010, <http://www.rfc-editor.org/info/rfc5973>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <http://www.rfc-editor.org/info/rfc6056>.

[RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <http://www.rfc-editor.org/info/rfc6092>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <http://www.rfc-editor.org/info/rfc6335>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC6396] Blunk, L., Karir, M., and C. Labovitz, "Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format", RFC 6396, DOI 10.17487/RFC6396, October 2011, <http://www.rfc-editor.org/info/rfc6396>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <http://www.rfc-editor.org/info/rfc6437>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <http://www.rfc-editor.org/info/rfc6438>.

[RFC6513] Rosen, E., Ed. and R. Aggarwal, Ed., "Multicast in MPLS/BGP IP VPNs", RFC 6513, DOI 10.17487/RFC6513, February 2012, <http://www.rfc-editor.org/info/rfc6513>.

[RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <http://www.rfc-editor.org/info/rfc6679>.

[RFC6726] Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport", RFC 6726, DOI 10.17487/RFC6726, November 2012, <http://www.rfc-editor.org/info/rfc6726>.

[RFC6773] Phelan, T., Fairhurst, G., and C. Perkins, "DCCP-UDP: A Datagram Congestion Control Protocol UDP Encapsulation for NAT Traversal", RFC 6773, DOI 10.17487/RFC6773, November 2012, <http://www.rfc-editor.org/info/rfc6773>.

[RFC6807] Farinacci, D., Shepherd, G., Venaas, S., and Y. Cai, "Population Count Extensions to Protocol Independent Multicast (PIM)", RFC 6807, DOI 10.17487/RFC6807, December 2012, <http://www.rfc-editor.org/info/rfc6807>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <http://www.rfc-editor.org/info/rfc6887>.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <http://www.rfc-editor.org/info/rfc6935>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <http://www.rfc-editor.org/info/rfc6936>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <http://www.rfc-editor.org/info/rfc6951>.

[RFC7143] Chadalapaka, M., Satran, J., Meth, K., and D. Black, "Internet Small Computer System Interface (iSCSI) Protocol (Consolidated)", RFC 7143, DOI 10.17487/RFC7143, April 2014, <http://www.rfc-editor.org/info/rfc7143>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

[RFC7450] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, DOI 10.17487/RFC7450, February 2015, <http://www.rfc-editor.org/info/rfc7450>.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <http://www.rfc-editor.org/info/rfc7510>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

[RFC7560] Kuehlewind, M., Ed., Scheffenegger, R., and B. Briscoe, "Problem Statement and Requirements for Increased Accuracy in Explicit Congestion Notification (ECN) Feedback", RFC 7560, DOI 10.17487/RFC7560, August 2015, <http://www.rfc-editor.org/info/rfc7560>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

[RFC7605] Touch, J., "Recommendations on Using Assigned Transport Port Numbers", BCP 165, RFC 7605, DOI 10.17487/RFC7605, August 2015, <http://www.rfc-editor.org/info/rfc7605>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <http://www.rfc-editor.org/info/rfc7657>.

[RFC7675] Perumal, M., Wing, D., Ravindranath, R., Reddy, T., and M. Thomson, "Session Traversal Utilities for NAT (STUN) Usage for Consent Freshness", RFC 7675, DOI 10.17487/RFC7675, October 2015, <http://www.rfc-editor.org/info/rfc7675>.

[STEVENS] Stevens, W., Fenner, B., and A. Rudoff, "UNIX Network Programming, The sockets Networking API", Addison-Wesley, 2004.

[UPnP] UPnP Forum, , "Internet Gateway Device (IGD) Standardized Device Control Protocol V 1.0", November 2001.

Appendix A. Case Study of the Use of IPv6 UDP Zero-Checksum Mode

This appendix provides a brief review of MPLS-in-UDP as an example of a UDP Tunnel Encapsulation that defines a UDP encapsulation. The purpose of the appendix is to provide a concrete example of which mechanisms were required in order to safely use UDP zero-checksum mode for MPLS-in-UDP tunnels over IPv6.

By default, UDP requires a checksum for use with IPv6. An option has been specified that permits a zero IPv6 UDP checksum when used in specific environments, specified in [RFC7510], and defines a set of operational constraints for use of this mode. These are summarized below:

A UDP tunnel or encapsulation using a zero-checksum mode with IPv6 must only be deployed within a single network (with a single network operator) or networks of an adjacent set of co-operating network operators where traffic is managed to avoid congestion, rather than over the Internet where congestion control is required. MPLS-in-UDP has been specified for networks under single administrative control (such as within a single operator’s network) where it is known (perhaps through knowledge of equipment types and lower layer checks) that packet corruption is exceptionally unlikely and where the operator is willing to take the risk of undetected packet corruption.

The tunnel encapsulator SHOULD use different IPv6 addresses for each UDP tunnel that uses the UDP zero-checksum mode, regardless of the decapsulator, to strengthen the decapsulator’s check of the IPv6 source address (i.e., the same IPv6 source address SHOULD NOT be used with more than one IPv6 destination address, independent of whether that destination address is a unicast or multicast address). Use of MPLS-in-UDP may be extended to networks within a set of closely cooperating network administrations (such as network operators who have agreed to work together to jointly provide specific services) [RFC7510].

The requirement for MPLS-in-UDP endpoints to check the source IPv6 address in addition to the destination IPv6 address, plus the strong recommendation against reuse of source IPv6 addresses among MPLS-in-UDP tunnels collectively provide some mitigation for the absence of UDP checksum coverage of the IPv6 header. In addition, the MPLS data plane only forwards packets with valid labels (i.e., labels that have been distributed by the tunnel egress Label Switched Router, LSR), providing some additional opportunity to detect MPLS-in-UDP packet misdelivery when the misdelivered packet contains a label that is not valid for forwarding at the receiving LSR. The expected result for IPv6 UDP zero-checksum mode for MPLS-in-UDP is that corruption of the destination IPv6 address will usually cause packet discard, as

offsetting corruptions to the source IPv6 and/or MPLS top label are unlikely.

Additional assurance is provided by the restrictions in the above exceptions that limit usage of IPv6 UDP zero-checksum mode to well-managed networks for which MPLS packet corruption has not been a problem in practice. Hence, MPLS-in-UDP is suitable for transmission over lower layers in well-managed networks that are allowed by the exceptions stated above and the rate of corruption of the inner IP packet on such networks is not expected to increase by comparison to MPLS traffic that is not encapsulated in UDP. For these reasons, MPLS-in-UDP does not provide an additional integrity check when UDP zero-checksum mode is used with IPv6, and this design is in accordance with requirements 2, 3 and 5 specified in Section 5 of [RFC6936].

The MPLS-in-UDP encapsulation does not provide a mechanism to safely fall back to using a checksum when a path change occurs that redirects a tunnel over a path that includes a middlebox that discards IPv6 datagrams with a zero UDP checksum. In this case, the MPLS-in-UDP tunnel will be black-holed by that middlebox. Recommended changes to allow firewalls, NATs and other middleboxes to support use of an IPv6 zero UDP checksum are described in Section 5 of [RFC6936]. MPLS does not accumulate incorrect state as a consequence of label stack corruption. A corrupt MPLS label results in either packet discard or forwarding (and forgetting) of the packet without accumulation of MPLS protocol state. Active monitoring of MPLS-in-UDP traffic for errors is REQUIRED because the occurrence of errors will result in some accumulation of error information outside the MPLS protocol for operational and management purposes. This design is in accordance with requirement 4 specified in Section 5 of [RFC6936]. In addition, IPv6 traffic with a zero UDP checksum MUST be actively monitored for errors by the network operator.

Operators SHOULD also deploy packet filters to prevent IPv6 packets with a zero UDP checksum from escaping from the network due to misconfiguration or packet errors. In addition, IPv6 traffic with a zero UDP checksum MUST be actively monitored for errors by the network operator.

Appendix B. Revision Notes

Note to RFC-Editor: please remove this entire section prior to publication.

Changes in draft-ietf-tsvwg-rfc5405bis-19:

- o Addressed IESG review comments.

Changes in draft-ietf-tsvwg-rfc5405bis-18:

- o Fix a nit.

Changes in draft-ietf-tsvwg-rfc5405bis-17:

- o Incorporated text from Mark Allman for the section on "Protocol Timer Guidelines".

Changes in draft-ietf-tsvwg-rfc5405bis-16:

- o Addressed suggestions by David Black.

Changes in draft-ietf-tsvwg-rfc5405bis-15:

- o Addressed more suggestions by Takeshi Takahashi.

Changes in draft-ietf-tsvwg-rfc5405bis-14:

- o Addressed SEC_DIR review by Takeshi Takahashi.
- o Addressed Gen-ART review by Paul Kyzivat.
- o Addressed OPS-DIR review by Tim Chown.
- o Addressed some of Mark Allman’s comments regarding RTTs and RTOs.
- o Addressed some of Brian Trammell;s comments regarding new IETF transports.

Changes in draft-ietf-tsvwg-rfc5405bis-13:

- o Minor corrections.
- o Changes recommended by Spencer Dawkins.
- o Placed the recommendations on timers within section 3.1
- o Updated the recommendations on reliability to also reference the recommendations on timers.

Changes in draft-ietf-tsvwg-rfc5405bis-12:

- o Introduced a separate section on the usage of timers to avoid repeating similar guidance in multiple sections.
- o Updated RTT measurement text to align with revised min RTO recommendation for TCP.

- o Updated text based on draft-ietf-tcpm-rto-consider-03 and to now cite this draft.

- o Fixed inconsistency in term used for keep-alive messages (keep-alive packet, keep-alives).

Changes in draft-ietf-tsvwg-rfc5405bis-11:

- o Address some issues that idnits found.

Changes in draft-ietf-tsvwg-rfc5405bis-10:

- o Restored changes from -08 that -09 accidentally rolled back.

Changes in draft-ietf-tsvwg-rfc5405bis-09:

- o Fix to cross reference in summary table.

Changes in draft-ietf-tsvwg-rfc5405bis-08:

This update introduces new text in the following sections:

- o The ID from RTGWG on encap Section 7 makes recommendations on entropy. Section 5.1 of the 5405bis draft had a single sentence on use of the UDP source port to inject entropy. Related work such as UDP-in-MPLS and GRE-in-UDP have also made recommendations on entropy usage. A new section has been added to address this.

- o Added reference to RFC2983 on DSCP with tunnels.

- o New text after comment from David Black on needing to improve the header protection text.

- o Replaced replace /controlled network environment/ with /controlled environment/ to be more consistent with other drafts.

- o Section 3.1.7 now explicitly refers to the applicability subsection describing controlled environments.

- o PLPMTUD section updated.

- o Reworded checksum text to place IPv6 UDP zero checksum text in a separate subsection (this became too long in the main section)

- o Updated summary table

Changes in draft-ietf-tsvwg-rfc5405bis-07:

This update introduces new text in the following sections:

- o Addressed David Black’s review during WG LC.

Changes in draft-ietf-tsvwg-rfc5405bis-06:

This update introduces new text in the following sections:

- o Multicast Congestion Control Guidelines (Section rewritten by Greg and Gorry to differentiate sender-driven and receiver-driven CC)
- o Using UDP Ports (Added a short para on RPF checks protecting from off-path attacks)
- o Applications using Multiple UDP Ports (Added text on layered multicast)

Changes in draft-ietf-tsvwg-rfc5405bis-05:

- o Amended text in section discussing RTT for CC (feedback from Colin)

Changes in draft-ietf-tsvwg-rfc5405bis-04:

- o Added text on consent freshness (STUN) - (From Colin)
- o Reworked text on ECN (From David)
- o Reworked text on RTT with CC (with help from Mirja)
- o Added references to [RFC7675], [I-D.ietf-rtgwg-dt-encap], [I-D.ietf-intarea-tunnels] and [RFC7510]

Changes in draft-ietf-tsvwg-rfc5405bis-03:

- o Mention crypto hash in addition to CRC for integrity protection. (From Magnus.)
- o Mention PCP. (From Magnus.)
- o More accurate text on secure RTP (From Magnus.)
- o Reordered abstract to reflect .bis focus (Gorry)
- o Added a section on ECN, with actual ECN requirements (Gorry, help from Mirja)
- o Added section on Implications of RTT on Congestion Control (Gorry)

- o Added note that this refers to other protocols over IP (E Nordmark, rtg encaps guidance)

- o Added reordering text between sessions (consistent with use of ECMP, rtg encaps guidance)

- o Reworked text on off-path data protection (port usage)

- o Updated summary table

Changes in draft-ietf-tsvwg-rfc5405bis-02:

- o Added note that guidance may be applicable beyond UDP to abstract (from Erik Nordmark).

- o Small editorial changes to fix English nits.

- o Added a circuit may provide benefit to CC tunnels by controlling envelope.

- o Added tunnels should ingress-filter by packet type (from Erik Nordmark).

- o Added tunnels should perform IETF ECN processing when supporting ECN.

- o Multicast apps may employ CC or a circuit breaker.

- o Added programming guidance on off-path attacks (with C. Perkins).

- o Added reference to ECN benefits.

Changes in draft-ietf-tsvwg-rfc5405bis-01:

- o Added text on DSCP-usage.

- o More guidance on use of the checksum, including an example of how MPLS/UDP allowed support of a zero IPv6 UDP Checksum in some cases.

- o Added description of diffuse usage.

- o Clarified usage of the source port field.

draft-eggert-tsvwg-rfc5405bis-01 was adopted by the TSVWG and resubmitted as draft-ietf-tsvwg-rfc5405bis-00. There were no technical changes.

Changes in draft-eggert-tsvwg-rfc5405bis-01:

- o Added Greg Shepherd as a co-author, based on the multicast guidelines that originated with him.

Changes in draft-eggert-tsvwg-rfc5405bis-00 (relative to RFC5405):

- o The words "application designers" were removed from the draft title and the wording of the abstract was clarified abstract.
- o New text to clarify various issues and set new recommendations not previously included in RFC 5405. These include new recommendations for multicast, the use of checksums with IPv6, ECMP, recommendations on port usage, use of ECN, use of DiffServ, circuit breakers (initial text), etc.

Authors' Addresses

Lars Eggert
NetApp
Sonnenallee 1
Kirchheim 85551
Germany

Phone: +49 151 120 55791
EMail: lars@netapp.com
URI: https://eggert.org/

Godred Fairhurst
University of Aberdeen
Department of Engineering
Fraser Noble Building
Aberdeen AB24 3UE
Scotland

EMail: gorry@erg.abdn.ac.uk
URI: http://www.erg.abdn.ac.uk/

Greg Shepherd
Cisco Systems
Tasman Drive
San Jose
USA

EMail: gjshep@gmail.com

Network WG
Internet-Draft
Expires: January 4, 2015
Intended Status: Standards Track
Updates: RFC 2872 (if accepted)

James Polk
Subha Dhesikan
Cisco Systems
July 4, 2014

Resource Reservation Protocol (RSVP) Application-ID Profiles for Voice and Video Streams
draft-ietf-tsvwg-rsvp-app-id-vv-profiles-02

Abstract

RFC 2872 defines an Resource Reservation Protocol (RSVP) object for application identifiers. This document uses that App-ID and gives implementers specific guidelines for differing voice and video stream identifications to nodes along a reservation path, creating specific profiles for voice and video session identification.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

RFC 2872 [RFC2872] describes the usage of policy elements for providing application information in Resource Reservation Protocol (RSVP) signaling [RFC2205]. The intention of providing this information is to enable application-based policy control. However, RFC 2872 does not enumerate any application profiles. The absence of explicit, uniform profiles leads to incompatible handling of these values and misapplied policies. An application profile used by a sender might not be understood by the intermediaries or receiver in a different domain. Therefore, there is a need to enumerate application profiles that are universally understood and applied for correct policy control.

Call control between endpoints has the ability to bind or associate many attributes to a reservation. One new attribute is currently being defined so as to establish the type of traffic contained in that reservation. This is accomplished via assigning a traffic label to the call (or session or flow) [ID-TRAF-CLASS].

This document takes the application traffic classes from [ID-TRAF-CLASS] and places those strings in the APP-ID object defined in RFC 2872. Thus, the intermediary devices (e.g., routers) processing the RSVP message can learn the identified profile within the Application-ID policy element for a particular reservation, and possibly be configured with the profile(s) to understand them

correctly, thus performing the correct admission control.

Another goal of this document is to the ability to signal an application profile which can then be translated into a DSCP value as per the choice of each domain. While the DCLASS object [RFC2996] allows the transfer of DSCP value in an RSVP message, that RFC does not allow the flexibility of having different domains choosing the DSCP value for the traffic classes that they maintain.

How these labels indicate the appropriate Differentiated Services Codepoint (DSCP) is out of scope for this document.

This document will break out each application type and propose how the values in application-id template should be populated for uniformity and interoperability.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC 2119].

2. RSVP Application ID Template

The template from RFC 2872 is as follows:

```
             0                   1                   2                   3
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    PE Length (8)              |   P-type = AUTH_APP           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Attribute Length           |   A-type =    |  Sub-type =   |
 |                               | POLICY_LOCATOR|   ASCII_DN    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 Application name as ASCII string              |
 |                         (e.g. SAP.EXE)                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

In line with how this policy element is constructed in RFC 2872, the A-type will remain "POLICY_LOCATOR".

The P-type field is first created in [RFC2752]. This document uses the existing P-type "AUTH_APP" for application traffic class.

The first Sub-type will be mandatory for every profile within this document, and will be "ASCII_DN". No other Sub-types are defined by any profile within this document, but MAY be included by individual implementations - and MUST be ignored if not understood by receiving implementations along the reservation path.

RFC 2872 states the #1 sub-element from RFC 2872 as the "identifier that uniquely identifies the application vendor", which is optional to include. This document modifies this vendor limitation so that the identifier need only be unique - and not limited to an application vendor (identifier). For example, this specification now allows an RFC that defines an industry recognizable term or string to be a valid identifier. For example, a term or string taken from another IETF document, such as "conversational" or "avconf" from [ID-TRAF-CLASS]. This sub-element is still optional to include.

The following subsections will define the values within the above template into specific profiles for voice and video identification.

3. The Voice and Video Application-ID Profiles

This section contains the elements of the Application ID policy object which is used to signal the application classes defined in [ID-TRAF-CLASS].

3.1 The Broadcast Profiles

Broadcast profiles are for minimally buffered one-way streaming flows, such as video surveillance, or Internet based concerts or non-VOD TV broadcasts such as live sporting events.

This document creates Broadcast profiles for

- Broadcast IPTV for audio and video
- Broadcast Live-events for audio and video
- Broadcast Surveillance for audio and video

Here is an example profile for identifying Broadcast Video-Surveillance

```
   AUTH_APP, POLICY_LOCATOR, ASCII_DN,
  "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
   APP=broadcast.video.surveillance, VER="
```

[Editor’s Note: "rfcXXXX" will be replaced with the RFC number assigned to the [ID-TRAF-CLASS] reference. This ’note’ should be removed during the RFC-Editor review process.]

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value at this time.

3.2 The Realtime Interactive Profiles

Realtime Interactive profiles are for on-line gaming, and both remote and virtual avconf applications, in which the timing is particularly important towards the feedback to uses of these applications. This traffic type will generally not be UDP based, with minimal tolerance to RTT delays.

This document creates Realtime Interactive profiles for

- Realtime-Interactive Gaming
- Realtime-Interactive Remote-Desktop
- Realtime-Interactive Virtualized-Desktop

Here is the profile for identifying Realtime-Interactive Gaming

```
   AUTH_APP, POLICY_LOCATOR, ASCII_DN,
  "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
   APP=realtime-interactive.gaming, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.3 The Multimedia Conferencing Profiles

There will be Multimedia Conferencing profiles for presentation data, application sharing and whiteboarding, where these applications will most often be associated with a larger Conversational (audio and/or audio/video) conference. Timing is important, but some minimal delays are acceptable, unlike the case for Realtime-Interactive traffic.

This document creates Multimedia-Conferencing profiles for

- Multimedia-Conferencing presentation-data
- Multimedia-Conferencing presentation-video
- Multimedia-Conferencing presentation-audio
- Multimedia-Conferencing application-sharing
- Multimedia-Conferencing whiteboarding

Here is the profile for identifying Multimedia-Conferencing Application-sharing

```
   AUTH_APP, POLICY_LOCATOR, ASCII_DN,
  "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
   APP=multimedia-conferencing.application-sharing, VER="
```

Where the Globally Unique Identifier (GUID) indicates the RFC reference that created this well-known string [ID-TRAF-CLASS], the

APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.4 The Multimedia Streaming Profiles

Multimedia Streaming profiles are for more significantly buffered one-way streaming flows than Broadcast profiles. These include...

This document creates Multimedia Streaming profiles for

- Multimedia-Streaming multiplex
- Multimedia-Streaming webcast

Here is the profile for identifying Multimedia Streaming webcast

```
   AUTH_APP, POLICY_LOCATOR, ASCII_DN,
  "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
   APP=multimedia-streaming.webcast, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.5 The Conversational Profiles

Conversational category is for realtime bidirectional communications, such as voice or video, and is the most numerous due to the choices of application with or without adjectives. The number of profiles is then doubled because there needs to be one for unadmitted and one for admitted. The IANA section lists all that are currently proposed for registration at this time, therefore there will not be an exhaustive list provided in this section.

This document creates Conversational profiles for

- Conversational Audio
- Conversational Audio Admitted
- Conversational Video
- Conversational Video Admitted
- Conversational Audio Avconf
- Conversational Audio Avconf Admitted
- Conversational Video Avconf
- Conversational Video Avconf Admitted
- Conversational Audio Immersive
- Conversational Audio Immersive Admitted
- Conversational Video Immersive
- Conversational Video Immersive Admitted

Here is an example profile for identifying Conversational Audio:

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.audio, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

4. Security considerations

The security considerations section within RFC 2872 sufficiently covers this document, with one possible exception - someone using the wrong template values (e.g., claiming a reservation is Multimedia Streaming when it is in fact Real-time Interactive). Given that each traffic flow is within separate reservations, and RSVP does not have the ability to police the type of traffic within any reservation, solving for this appears to be administratively handled at best. This is not meant to be a ’punt’, but there really is nothing this template creates that is going to make things any harder for anyone (that we know of now).

5. IANA considerations

5.1 Application Profiles

This document requests IANA create a new registry for the application identification classes similar to the following table within the Resource Reservation Protocol (RSVP) Parameters registry:

Registry Name: RSVP APP-ID Profiles
Reference: [this document]
Registration procedures: Standards Track document [RFC5226]

[Editor’s Note: "rfcXXXX" will be replaced with the RFC number assigned to the [ID-TRAF-CLASS] reference. This ’note’ should be removed during the RFC-Editor review process.]

5.1.1 Broadcast Profiles IANA Registry

Broadcast Audio IPTV Profile

```
P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
                   "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                    APP=broadcast.audio.iptv, VER="
Reference: [this document]
```

```
   Broadcast Video IPTV Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=broadcast.video.iptv, VER="
     Reference: [this document]

   Broadcast Audio Live-events Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=broadcast.audio.live-events, VER="
     Reference: [this document]

   Broadcast Video Live-events Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=broadcast.video.live-events, VER="
     Reference: [this document]

   Broadcast Audio-Surveillance Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=broadcast.audio.surveillance, VER="
     Reference: [this document]

   Broadcast Video-Surveillance Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=broadcast.video.surveillance, VER="
     Reference: [this document]
```

5.1.2 Realtime-Interactive Profiles IANA Registry

```
   Realtime-Interactive Gaming Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
```

```
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP= realtime-interactive.gaming, VER="
     Reference: [this document]

   Real-time Interactive Remote-Desktop Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=realtime-interactive.remote-desktop, VER="
     Reference: [this document]

   Real-time Interactive Virtualized-Desktop Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=realtime-interactive.
                            remote-desktop.virtual, VER="
     Reference: [this document]

   Real-time Interactive Telemetry Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=realtime-interactive.telemetry, VER="
     Reference: [this document]
```

5.1.3 Multimedia-Conferencing Profiles IANA Registry

```
   Multimedia-Conferencing Presentation-Data Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                      "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                       APP= multimedia-conferencing.presentation-data,
                       VER="
     Reference: [this document]

   Multimedia-Conferencing Presentation-Video Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                      "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
```

```
                        APP= multimedia-conferencing.presentation-video,
                        VER="
     Reference: [this document]

   Multimedia-Conferencing Presentation-Audio Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                      "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                       APP= multimedia-conferencing.presentation-audio,
                       VER="
     Reference: [this document]

   Multimedia-Conferencing Application-Sharing Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                     "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                      APP= multimedia-conferencing.application-sharing,
                      VER="
     Reference: [this document]

   Multimedia-Conferencing Whiteboarding Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                     "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                      APP= multimedia-conferencing.whiteboarding, VER="
     Reference: [this document]
```

5.1.4 Multimedia-Streaming Profiles IANA Registry

```
   Multimedia-Streaming Multiplex Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=multimedia-streaming.multiplex, VER="
     Reference: [this document]

   Multimedia-Streaming Webcast Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=multimedia-streaming.webcast, VER="
```

```
     Reference: [this document]
```

5.1.5 Conversational Profiles IANA Registry

```
   Conversational Audio Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.audio, VER="
     Reference: [this document]

   Conversational Audio Admitted Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.audio.aq:admitted, VER="
     Reference: [this document]

   Conversational Video Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.video, VER="
     Reference: [this document]

   Conversational Video Admitted Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.video.aq:admitted, VER="
     Reference: [this document]

   Conversational Audio Avconf Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.audio.avconf, VER="
     Reference: [this document]

   Conversational Audio Avconf Admitted Profile
     P-type = AUTH_APP
```

```
A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                     APP=conversational.audio.avconf.aq:admitted,
                     VER="
  Reference: [this document]

Conversational Video Avconf Profile
  P-type = AUTH_APP
  A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                     APP=conversational.video.avconf, VER="
  Reference: [this document]

Conversational Video Avconf Admitted Profile
  P-type = AUTH_APP
  A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                     APP=conversational.video.avconf.aq:admitted,
                     VER="
  Reference: [this document]

Conversational Audio Immersive Profile
  P-type = AUTH_APP
  A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                     APP=conversational.audio.immersive, VER="
  Reference: [this document]

Conversational Audio Immersive Admitted Profile
  P-type = AUTH_APP
  A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                     APP=conversational.audio.immersive.aq:admitted,
                     VER="
  Reference: [this document]

Conversational Video Immersive Profile
  P-type = AUTH_APP
  A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
```

```
                        APP=conversational.video.immersive, VER="
     Reference: [this document]

   Conversational Video Immersive Admitted Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.video.immersive.aq:admitted,
                        VER="
     Reference: [this document]
```

6. Acknowledgments

To Francois Le Faucheur, Paul Jones, Ken Carlberg, Georgios Karagiannis and Glen Lavers for their helpful comments, document reviews and encouragement.

7. References

7.1. Normative References

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997
Functional Specification", RFC 2205, September 1997

[RFC2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997

[RFC2474] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers ", RFC 2474, December 1998

[RFC2750] S. Herzog, "RSVP Extensions for Policy Control", RFC 2750, January 2000

[RFC2872] Y. Bernet, R. Pabbati, "Application and Sub Application Identity Policy Element for Use with RSVP", RFC 2872, June 2000

[RFC2996] Y. Bernet, "Format of the RSVP DCLASS Object", RFC 2996, November 2000

[RFC3182] S. Yadav, R. Yavatkar, R. Pabbati, P. Ford, T. Moore, S. Herzog, R. Hess, "Identity Representation for RSVP", RFC 3182, October 2001

[RFC5226] T. Narten, H. Alvestrand, "Guidelines for Writing an IANA

Considerations Section in RFCs", RFC 5226, May 2008

[ID-TRAF-CLASS] J. Polk, S. Dhesikan, P. Jones, "The Session Description Protocol (SDP) 'trafficclass' Attribute", work in progress, Feb 2013

7.2. Informative References

[RFC4594] J. Babiarz, K. Chan, F Baker, "Configuration Guidelines for Diffserv Service Classes", RFC 4594, August 2006

Authors' Addresses

James Polk
3913 Treemont Circle
Colleyville, Texas, USA
+1.817.271.3552

mailto: jmpolk@cisco.com

Subha Dhesikan
170 W Tasman St
San Jose, CA, USA
+1.408-902-3351

mailto: sdhesika@cisco.com

Appendix - Changes to ID

[Editor's Note: this appendix should be removed in the RFC-Editor's process.]

A.1 - Changes from WG version -00 to WG version -01

The following changes were made in this version:

- corrected nits

- globally replaced GUID link from the MMUSIC Trafficclass ID to the future RFC of that document.

- added profiles for presentation-video and presentation-audio

A.2 - Changes from Individual -04 to WG version -00

The following changes were made in this version:

- changed P-Type from APP_TC back to AUTH_APP, which is already defined.

- fixed nits and inconsistencies

A.3 - Changes from Individual -03 to -04

The following changes were made in this version:

- clarified security considerations section to mean RSVP cannot police the type of traffic within a reservation to know if a traffic flow should be using a different profile, as defined in this document.

- changed existing informative language regarding "... other Sub-types ..." from 'can' to normative 'MAY'.
- editorial changes to clear up minor mistakes

A.4 - Changes from Individual -02 to -03

The following changes were made in this version:

- Added [ID-TRAF-CLASS] as a reference

- Changed to a new format of the profile string.

- Added many new profiles based on the new format into each parent category of Section 3.

- changed the GUID to refer to draft-ietf-mmusic-traffic-class-for-sdp-03.txt

- changed 'desktop' adjective to 'avconf' to keep in alignment with [ID-TRAF-CLASS]

- Have a complete IANA Registry proposal for each application-ID discussed in this draft.

- General text clean-up of the draft.

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 20, 2017

P. Jones
S. Dhesikan
C. Jennings
Cisco Systems
D. Druta
AT&T

August 19, 2016

DSCP Packet Markings for WebRTC QoS
draft-ietf-tsvwg-rtcweb-qos-18

Abstract

Many networks, such as service provider and enterprise networks, can provide different forwarding treatments for individual packets based on Differentiated Services Code Point (DSCP) values on a per-hop basis. This document provides the recommended DSCP values for web browsers to use for various classes of WebRTC traffic.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Differentiated Services Code Point (DSCP) [RFC2474] packet marking can help provide QoS in some environments. This specification provides default packet marking for browsers that support WebRTC applications, but does not change any advice or requirements in other IETF RFCs. The contents of this specification are intended to be a simple set of implementation recommendations based on the previous RFCs.

Networks where these DSCP markings are beneficial (likely to improve QoS for WebRTC traffic) include:

1. Private, wide-area networks. Network administrators have control over remarking packets and treatment of packets.

2. Residential Networks. If the congested link is the broadband uplink in a cable or DSL scenario, often residential routers/NAT support preferential treatment based on DSCP.

3. Wireless Networks. If the congested link is a local wireless network, marking may help.

There are cases where these DSCP markings do not help, but, aside from possible priority inversion for "less than best effort traffic"

(see Section 5), they seldom make things worse if packets are marked appropriately.

DSCP values are in principle site specific, with each site selecting its own code points for controlling per-hop-behavior to influence the QoS for transport-layer flows. However in the WebRTC use cases, the browsers need to set them to something when there is no site specific information. This document describes a subset of DSCP code point values drawn from existing RFCs and common usage for use with WebRTC applications. These code points are intended to be the default values used by a WebRTC application. While other values could be used, using a non-default value may result in unexpected per-hop behavior. It is RECOMMENDED that WebRTC applications use non-default values only in private networks that are configured to use different values.

This specification defines inputs that are provided by the WebRTC application hosted in the browser that aid the browser in determining how to set the various packet markings. The specification also defines the mapping from abstract QoS policies (flow type, priority level) to those packet markings.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The terms "browser" and "non-browser" are defined in [RFC7742] and carry the same meaning in this document.

3. Relation to Other Specifications

This document is a complement to [RFC7657], which describes the interaction between DSCP and real-time communications. That RFC covers the implications of using various DSCP values, particularly focusing on Real-time Transport Protocol (RTP) [RFC3550] streams that are multiplexed onto a single transport-layer flow.

There are a number of guidelines specified in [RFC7657] that apply to marking traffic sent by WebRTC applications, as it is common for multiple RTP streams to be multiplexed on the same transport-layer flow. Generally, the RTP streams would be marked with a value as appropriate from Table 1. A WebRTC application might also multiplex data channel [I-D.ietf-rtcweb-data-channel] traffic over the same 5-tuple as RTP streams, which would also be marked as per that table. The guidance in [RFC7657] says that all data channel traffic would be marked with a single value that is typically different than the

value(s) used for RTP streams multiplexed with the data channel traffic over the same 5-tuple, assuming RTP streams are marked with a value other than default forwarding (DF). This is expanded upon further in the next section.

This specification does not change or override the advice in any other IETF RFCs about setting packet markings. Rather, it simply selects a subset of DSCP values that is relevant in the WebRTC context.

The DSCP value set by the endpoint is not trusted by the network. In addition, the DSCP value may be remarked at any place in the network for a variety of reasons to any other DSCP value, including default forwarding (DF) value to provide basic best effort service. Even so, there is benefit in marking traffic even if it only benefits the first few hops. The implications are discussed in Secton 3.2 of [RFC7657]. Further, a mitigation for such action is through an authorization mechanism. Such an authorization mechanism is outside the scope of this document.

4. Inputs

WebRTC applications send and receive two types of flows of significance to this document:

- o media flows which are RTP streams [I-D.ietf-rtcweb-rtp-usage]

- o data flows which are data channels [I-D.ietf-rtcweb-data-channel]

Each of the RTP streams and distinct data channels consists of all of the packets associated with an independent media entity, so an RTP stream or distinct data channel is not always equivalent to a transport-layer flow defined by a 5-tuple (source address, destination address, source port, destination port, and protocol). There may be multiple RTP streams and data channels multiplexed over the same 5-tuple, with each having a different level of importance to the application and, therefore, potentially marked using different DSCP values than another RTP stream or data channel within the same transport-layer flow. (Note that there are restrictions with respect to marking different data channels carried within the same SCTP association as outlined in Section 5.)

The following are the inputs provided by the WebRTC application to the browser:

- o Flow Type: The application provides this input because it knows if the flow is audio, interactive video [RFC4594] [G.1010] with or without audio, or data.

- o Application Priority: Another input is the relative importance of an RTP stream or data channel. Many applications have multiple flows of the same Flow Type and often some flows are more important than others. For example, in a video conference where there are usually audio and video flows, the audio flow may be more important than the video flow. JavaScript applications can tell the browser whether a particular flow is high, medium, low or very low importance to the application.

[I-D.ietf-rtcweb-transports] defines in more detail what an individual flow is within the WebRTC context and priorities for media and data flows.

Currently in WebRTC, media sent over RTP is assumed to be interactive [I-D.ietf-rtcweb-transports] and browser APIs do not exist to allow an application to to differentiate between interactive and non-interactive video.

5. DSCP Mappings

The DSCP values for each flow type of interest to WebRTC based on application priority are shown in Table 1. These values are based on the framework and recommended values in [RFC4594]. A web browser SHOULD use these values to mark the appropriate media packets. More information on EF can be found in [RFC3246]. More information on AF can be found in [RFC2597]. DF is default forwarding which provides the basic best effort service [RFC2474].

WebRTC use of multiple DSCP values may encounter network blocking of packets with certain DSCP values. See section 4.2 of [I-D.ietf-rtcweb-transports] for further discussion, including how WebRTC implementations establish and maintain connectivity when such blocking is encountered.

Flow Type	Very Low	Low	Medium	High
Audio	CS1 (8)	DF (0)	EF (46)	EF (46)
Interactive Video with or without Audio	CS1 (8)	DF (0)	AF42, AF43 (36, 38)	AF41, AF42 (34, 36)
Non-Interactive Video with or without Audio	CS1 (8)	DF (0)	AF32, AF33 (28, 30)	AF31, AF32 (26, 28)
Data	CS1 (8)	DF (0)	AF11	AF21

Table 1: Recommended DSCP Values for WebRTC Applications

The application priority, indicated by the columns "very low", "low", "Medium", and "high", signifies the relative importance of the flow within the application. It is an input that the browser receives to assist in selecting the DSCP value and adjusting the network transport behavior.

The above table assumes that packets marked with CS1 are treated as "less than best effort", such as the LE behavior described in [RFC3662]. However, the treatment of CS1 is implementation dependent. If an implementation treats CS1 as other than "less than best effort", then the actual priority (or, more precisely, the per-hop-behavior) of the packets may be changed from what is intended. It is common for CS1 to be treated the same as DF, so applications and browsers using CS1 cannot assume that CS1 will be treated differently than DF [RFC7657]. However, it is also possible per [RFC2474] for CS1 traffic to be given better treatment than DF, thus caution should be exercised when electing to use CS1. This is one of the cases where marking packets using these recommendations can make things worse.

Implementers should also note that excess EF traffic is dropped. This could mean that a packet marked as EF may not get through, although the same packet marked with a different DSCP value would have gotten through. This is not a flaw, but how excess EF traffic is intended to be treated.

The browser SHOULD first select the flow type of the flow. Within the flow type, the relative importance of the flow SHOULD be used to select the appropriate DSCP value.

Currently, all WebRTC video is assumed to be interactive [I-D.ietf-rtcweb-transports], for which the Interactive Video DSCP values in Table 1 SHOULD be used. Browsers MUST NOT use the AF3x DSCP values (for Non-Interactive Video in Table 1) for WebRTC applications. Non-browser implementations of WebRTC MAY use the AF3x DSCP values for video that is known not to be interactive, e.g., all video in a WebRTC video playback application that is not implemented in a browser.

The combination of flow type and application priority provides specificity and helps in selecting the right DSCP value for the flow. All packets within a flow SHOULD have the same application priority. In some cases, the selected application priority cell may have multiple DSCP values, such as AF41 and AF42. These offer different drop precedences. The different drop precedence values provides additional granularity in classifying packets within a flow. For example, in a video conference the video flow may have medium application priority, thus either AF42 or AF43 may be selected. More important video packets (e.g., a video picture or frame encoded without any dependency on any prior pictures or frames) might be marked with AF42 and less important packets (e.g., a video picture or frame encoded based on the content of one or more prior pictures or frames) might be marked with AF43 (e.g., receipt of the more important packets enables a video renderer to continue after one or more packets are lost).

It is worth noting that the application priority is utilized by the coupled congestion control mechanism for media flows per [I-D.ietf-rmcat-coupled-cc] and the SCTP scheduler for data channel traffic per [I-D.ietf-rtcweb-data-channel].

For reasons discussed in Section 6 of [RFC7657], if multiple flows are multiplexed using a reliable transport (e.g., TCP) then all of the packets for all flows multiplexed over that transport-layer flow MUST be marked using the same DSCP value. Likewise, all WebRTC data channel packets transmitted over an SCTP association MUST be marked using the same DSCP value, regardless of how many data channels (streams) exist or what kind of traffic is carried over the various SCTP streams. In the event that the browser wishes to change the DSCP value in use for an SCTP association, it MUST reset the SCTP congestion controller after changing values. Frequent changes in the DSCP value used for an SCTP association are discouraged, though, as this would defeat any attempts at effectively managing congestion. It should also be noted that any change in DSCP value that results in a reset of the congestion controller puts the SCTP association back into slow start, which may have undesirable effects on application performance.

For the data channel traffic multiplexed over an SCTP association, it is RECOMMENDED that the DSCP value selected be the one associated with the highest priority requested for all data channels multiplexed over the SCTP association. Likewise, when multiplexing multiple flows over a TCP connection, the DCSP value selected should be the one associated with the highest priority requested for all multiplexed flows.

If a packet enters a network that has no support for a flow type-application priority combination specified in Table 1, then the network node at the edge will remark the DSCP value based on policies. This could result in the flow not getting the network treatment it expects based on the original DSCP value in the packet. Subsequently, if the packet enters a network that supports a larger number of these combinations, there may not be sufficient information in the packet to restore the original markings. Mechanisms for restoring such original DSCP is outside the scope of this document.

In summary, DSCP marking provides neither guarantees nor promised levels of service. However, DSCP marking is expected to provide a statistical improvement in real-time service as a whole. The service provided to a packet is dependent upon the network design along the path, as well as the network conditions at every hop.

6. Security Considerations

Since the JavaScript application specifies the flow type and application priority that determine the media flow DSCP values used by the browser, the browser could consider application use of a large number of higher priority flows to be suspicious. If the server hosting the JavaScript application is compromised, many browsers within the network might simultaneously transmit flows with the same DSCP marking. The DiffServ architecture requires ingress traffic conditioning for reasons that include protecting the network from this sort of attack.

Otherwise, this specification does not add any additional security implications beyond those addressed in the following DSCP-related specifications. For security implications on use of DSCP, please refer to Section 7 of [RFC7657] and Section 6 of [RFC4594]. Please also see [I-D.ietf-rtcweb-security] as an additional reference.

7. IANA Considerations

This specification does not require any actions from IANA.

8. Downward References

This specification contains a downwards reference to [RFC4594] and [RFC7657]. However, the parts of the former RFC used by this specification are sufficiently stable for this downward reference. The guidance in the latter RFC is necessary to understand the Diffserv technology used in this document and the motivation for the recommended DSCP values and procedures.

9. Acknowledgements

Thanks to David Black, Magnus Westerlund, Paolo Severini, Jim Hasselbrook, Joe Marcus, Erik Nordmark, Michael Tuexen, and Brian Carpenter for their invaluable input.

10. Dedication

This document is dedicated to the memory of James Polk, a long-time friend and colleague. James made important contributions to this specification, including serving initially as one of the primary authors. The IETF global community mourns his loss and he will be missed dearly.

11. Document History

Note to RFC Editor: Please remove this section.

This document was originally an individual submission in RTCWeb WG. The RTCWeb working group selected it to be become a WG document. Later the transport ADs requested that this be moved to the TSVWG WG as that seemed to be a better match.

12. References

12.1. Normative References

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ietf-rtcweb-rtp-usage]
Perkins, D., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-26 (work in progress), March 2016.

[I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", draft-ietf-rtcweb-security-08 (work in progress), February 2015.

[I-D.ietf-rtcweb-transports]
Alvestrand, H., "Transports for WebRTC", draft-ietf-rtcweb-transports-15 (work in progress), August 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <http://www.rfc-editor.org/info/rfc4594>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <http://www.rfc-editor.org/info/rfc7657>.

[RFC7742] Roach, A., "WebRTC Video Processing and Codec Requirements", RFC 7742, DOI 10.17487/RFC7742, March 2016, <http://www.rfc-editor.org/info/rfc7742>.

12.2. Informative References

[G.1010] International Telecommunications Union, "End-user multimedia QoS categories", Recommendation ITU-T G.1010, November 2001.

[I-D.ietf-rmcat-coupled-cc]
Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-03 (work in progress), July 2016.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <http://www.rfc-editor.org/info/rfc2474>.

[RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, DOI 10.17487/RFC2597, June 1999, <http://www.rfc-editor.org/info/rfc2597>.

[RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, DOI 10.17487/RFC3246, March 2002, <http://www.rfc-editor.org/info/rfc3246>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services", RFC 3662, DOI 10.17487/RFC3662, December 2003, <http://www.rfc-editor.org/info/rfc3662>.

Authors' Addresses

Paul E. Jones
Cisco Systems

Email: paulej@packetizer.com

Subha Dhesikan
Cisco Systems

Email: sdhesika@cisco.com

Cullen Jennings
Cisco Systems

Email: fluffy@cisco.com

Dan Druta
AT&T

Email: dd5826@att.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 28, 2015

M. Tuexen
Muenster Univ. of Appl. Sciences
R. Stewart
Netflix, Inc.
R. Jesup
WorldGate Communications
S. Loreto
Ericsson

January 24, 2015

DTLS Encapsulation of SCTP Packets
draft-ietf-tsvwg-sctp-dtls-encaps-09.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a transport protocol originally defined to run on top of the network protocols IPv4 or IPv6. This document specifies how SCTP can be used on top of the Datagram Transport Layer Security (DTLS) protocol. Using the encapsulation method described in this document, SCTP is unaware of the protocols being used below DTLS; hence explicit IP addresses cannot be used in the SCTP control chunks. As a consequence, the SCTP associations carried over DTLS can only be single homed.

Status of This Memo

Copyright Notice

Table of Contents

1. Overview

The Stream Control Transmission Protocol (SCTP) as defined in [RFC4960] is a transport protocol running on top of the network protocols IPv4 [RFC0791] or IPv6 [RFC2460]. This document specifies how SCTP is used on top of the Datagram Transport Layer Security (DTLS) protocol. DTLS 1.0 is defined in [RFC4347] and the latest version when this RFC was published, DTLS 1.2, is defined in [RFC6347]. This encapsulation is used for example within the WebRTC protocol suite (see [I-D.ietf-rtcweb-overview] for an overview) for transporting non-SRTP data between browsers. The architecture of this stack is described in [I-D.ietf-rtcweb-data-channel].

[NOTE to RFC-Editor:

Please ensure that the authors double check the above statement about DTLS 1.2 during AUTH48 and then remove this note before publication.

]

```
+----------+
|   SCTP   |
+----------+
|   DTLS   |
+----------+
| ICE/UDP  |
+----------+
```

Figure 1: Basic stack diagram

This encapsulation of SCTP over DTLS over UDP or ICE/UDP (see [RFC5245]) can provide a NAT traversal solution in addition to confidentiality, source authentication, and integrity protected transfers. Please note that using ICE does not necessarily imply that a different packet format is used on the wire.

Please note that the procedures defined in [RFC6951] for dealing with the UDP port numbers do not apply here. When using the encapsulation defined in this document, SCTP is unaware about the protocols used below DTLS.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Encapsulation and Decapsulation Procedure

When an SCTP packet is provided to the DTLS layer, the complete SCTP packet, consisting of the SCTP common header and a number of SCTP chunks, is handled as the payload of the application layer protocol of DTLS. When the DTLS layer has processed a DTLS record containing a message of the application layer protocol, the payload is passed to the SCTP layer. The SCTP layer expects an SCTP common header followed by a number of SCTP chunks.

4. General Considerations

An implementation of SCTP over DTLS MUST implement and use a path maximum transmission unit (MTU) discovery method that functions without ICMP to provide SCTP/DTLS with an MTU estimate. An implementation of "Packetization Layer Path MTU Discovery" [RFC4821] either in SCTP or DTLS is RECOMMENDED.

The path MTU discovery is performed by SCTP when SCTP over DTLS is used for data channels (see Section 5 of [I-D.ietf-rtcweb-data-channel]).

5. DTLS Considerations

The DTLS implementation MUST support DTLS 1.0 [RFC4347] and SHOULD support the most recently published version of DTLS, which was DTLS 1.2 [RFC6347] when this RFC was published. In the absence of a revision to this document, the latter requirement applies to all future versions of DTLS when they are published as RFCs. This document will only be revised if a revision to DTLS or SCTP makes a revision to the encapsulation necessary.

[NOTE to RFC-Editor:

Please ensure that the authors double check the above statement about DTLS 1.2 during AUTH48 and then remove this note before publication.

]

SCTP performs segmentation and reassembly based on the path MTU. Therefore the DTLS layer MUST NOT use any compression algorithm.

The DTLS MUST support sending messages larger than the current path MTU. This might result in sending IP level fragmented messages.

If path MTU discovery is performed by the DTLS layer, the method described in [RFC4821] MUST be used. For probe packets, the extension defined in [RFC6520] MUST be used.

If path MTU discovery is performed by the SCTP layer and IPv4 is used as the network layer protocol, the DTLS implementation SHOULD allow the DTLS user to enforce that the corresponding IPv4 packet is sent with the Don’t Fragment (DF) bit set. If controlling the DF bit is not possible, for example due to implementation restrictions, a safe value for the path MTU has to be used by the SCTP stack. It is RECOMMENDED that the safe value does not exceed 1200 bytes. Please note that [RFC1122] only requires end hosts to be able to reassemble fragmented IP packets up to 576 bytes in length.

The DTLS implementation SHOULD allow the DTLS user to set the Differentiated services code point (DSCP) used for IP packets being sent (see [RFC2474]). This requires the DTLS implementation to pass the value through and the lower layer to allow setting this value. If the lower layer does not support setting the DSCP, then the DTLS user will end up with the default value used by protocol stack. Please note that only a single DSCP value can be used for all packets belonging to the same SCTP association.

Using explicit congestion notifications (ECN) in SCTP requires the DTLS layer to pass the ECN bits through and its lower layer to expose access to them for sent and received packets (see [RFC3168]). The implementation of DTLS and its lower layer have to provide this support. If this is not possible, for example due to implementation restrictions, ECN can't be used by SCTP.

6. SCTP Considerations

This section describes the usage of the base protocol and the applicability of various SCTP extensions.

6.1. Base Protocol

This document uses SCTP [RFC4960] with the following restrictions, which are required to reflect that the lower layer is DTLS instead of IPv4 and IPv6 and that SCTP does not deal with the IP addresses or the transport protocol used below DTLS:

- o A DTLS connection MUST be established before an SCTP association can be set up.

- o Multiple SCTP associations MAY be multiplexed over a single DTLS connection. The SCTP port numbers are used for multiplexing and demultiplexing the SCTP associations carried over a single DTLS connection.

- o All SCTP associations are single-homed, because DTLS does not expose any address management to its upper layer. Therefore it is RECOMMENDED to set the SCTP parameter path.max.retrans to association.max.retrans.

- o The INIT and INIT-ACK chunk MUST NOT contain any IPv4 Address or IPv6 Address parameters. The INIT chunk MUST NOT contain the Supported Address Types parameter.

- o The implementation MUST NOT rely on processing ICMP or ICMPv6 packets, since the SCTP layer most likely is unable to access the SCTP common header in the plain text of the packet, which triggered the sending of the ICMP or ICMPv6 packet. This applies in particular to path MTU discovery when performed by SCTP.

- o If the SCTP layer is notified about a path change by its lower layers, SCTP SHOULD retest the Path MTU and reset the congestion state to the initial state. The window-based congestion control method specified in [RFC4960], resets the congestion window and slow start threshold to their initial values.

6.2. Padding Extension

When the SCTP layer performs path MTU discovery as specified in [RFC4821], the padding extension defined in [RFC4820] MUST be supported and used for probe packets (HEARTBEAT chunks bundled with PADDING chunks [RFC4820]).

6.3. Dynamic Address Reconfiguration Extension

If the dynamic address reconfiguration extension defined in [RFC5061] is used, ASCONF chunks MUST use wildcard addresses only.

6.4. SCTP Authentication Extension

The SCTP authentication extension defined in [RFC4895] can be used with DTLS encapsulation, but does not provide any additional benefit.

6.5. Partial Reliability Extension

Partial reliability as defined in [RFC3758] can be used in combination with DTLS encapsulation. It is also possible to use additional PR-SCTP policies, for example the ones defined in [I-D.ietf-tsvwg-sctp-prpolicies].

6.6. Stream Reset Extension

The SCTP stream reset extension defined in [RFC6525] can be used with DTLS encapsulation. It is used to reset SCTP streams and add SCTP streams during the lifetime of the SCTP association.

6.7. Interleaving of Large User Messages

SCTP as defined in [RFC4960] does not support the interleaving of large user messages that need to be fragmented and reassembled by the SCTP layer. The protocol extension defined in [I-D.ietf-tsvwg-sctp-ndata] overcomes this limitation and can be used with DTLS encapsulation.

7. IANA Considerations

This document requires no actions from IANA.

8. Security Considerations

Security considerations for DTLS are specified in [RFC4347] and for SCTP in [RFC4960], [RFC3758], and [RFC6525]. The combination of SCTP and DTLS introduces no new security considerations.

SCTP should not process the IP addresses used for the underlying communication since DTLS provides no guarantees about them.

It should be noted that the inability to process ICMP or ICMPv6 messages does not add any security issue. When SCTP is carried over a connection-less lower layer like IPv4, IPv6, or UDP, processing of these messages is required to protect other nodes not supporting SCTP. Since DTLS provides a connection-oriented lower layer, this kind of protection is not necessary.

9. Acknowledgments

The authors wish to thank David Black, Benoit Claise, Spencer Dawkins, Francis Dupont, Gorry Fairhurst, Stephen Farrell, Christer Holmberg, Barry Leiba, Eric Rescorla, Tom Taylor, Joe Touch and Magnus Westerlund for their invaluable comments.

10. References

10.1. Normative References

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, March 2007.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.

10.2. Informative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, August 2007.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, February 2012.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013.

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-13 (work in progress), November 2014.

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ietf-tsvwg-sctp-prpolicies]
Tuexen, M., Seggelmann, R., Stewart, R., and S. Loreto, "Additional Policies for the Partial Reliability Extension of the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-prpolicies-06 (work in progress), December 2014.

[I-D.ietf-tsvwg-sctp-ndata]
Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "Stream Schedulers and a New Data Chunk for the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-ndata-02 (work in progress), January 2015.

Appendix A. NOTE to the RFC-Editor

Although the references to [I-D.ietf-tsvwg-sctp-prpolicies] and [I-D.ietf-tsvwg-sctp-ndata] are informative, put this document in REF-HOLD until these two references have been approved and update these references to the corresponding RFCs.

Authors' Addresses

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
DE

Email: tuexen@fh-muenster.de

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
US

Email: randall@lakerest.net

Randell Jesup
WorldGate Communications
3800 Horizon Blvd, Suite #103
Trevose, PA 19053-4947
US

Phone: +1-215-354-5166
Email: randell_ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: Salvatore.Loreto@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 20, 2016

Y. Nishida
GE Global Research
P. Natarajan
Cisco Systems
A. Caro
BBN Technologies
P. Amer
University of Delaware
K. Nielsen
Ericsson

February 17, 2016

SCTP-PF: Quick Failover Algorithm in SCTP
draft-ietf-tsvwg-sctp-failover-16.txt

Abstract

SCTP supports multi-homing. However, when the failover operation specified in RFC4960 is followed, there can be significant delay and performance degradation in the data transfer path failover. To overcome this problem this document specifies a quick failover algorithm (SCTP-PF) based on the introduction of a Potentially Failed (PF) state in SCTP Path Management.

The document also specifies a dormant state operation of SCTP. This dormant state operation is required to be followed by an SCTP-PF implementation, but it may equally well be applied by a standard RFC4960 SCTP implementation.

Additionally, the document introduces an alternative switchback operation mode called Primary Path Switchover that will be beneficial in certain situations. This mode of operation applies to both a standard RFC4960 SCTP implementation as well as to a SCTP-PF implementation.

The procedures defined in the document require only minimal modifications to the RFC4960 specification. The procedures are sender-side only and do not impact the SCTP receiver.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Stream Control Transmission Protocol (SCTP) specified in [RFC4960] supports multi-homing at the transport layer. SCTP's multi-homing features include failure detection and failover procedures to provide network interface redundancy and improved end-to-end fault tolerance. In SCTP's current failure detection procedure, the sender must experience Path.Max.Retrans (PMR) number of consecutive failed timer-based retransmissions on a destination address before detecting a path failure. Until detecting the path failure, the sender continues to transmit data on the failed path. The prolonged time in which [RFC4960] SCTP continues to use a failed path severely degrades the performance of the protocol. To address this problem, this document specifies a quick failover algorithm (SCTP-PF) based on the introduction of a new Potentially Failed (PF) path state in SCTP path management. The performance deficiencies of the [RFC4960] failover operation, and the improvements obtainable from the introduction of a Potentially Failed state in SCTP, were proposed and documented in [NATARAJAN09] for Concurrent Multipath Transfer SCTP [IYENGAR06].

While SCTP-PF can accelerate failover process and improve performance, the risks that an SCTP endpoint enters the dormant state where all destination addresses are inactive can be increased. [RFC4960] leaves the protocol operation during dormant state to implementations and encourages to avoid entering the state as much as possible by careful tuning of the Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) parameters. We specify a dormant state operation for SCTP-PF which makes SCTP-PF provide the same disruption tolerance as [RFC4960] despite that the dormant state may be entered more quickly. The dormant state operation may equally well be applied by an [RFC4960] implementation and will here serve to provide added fault tolerance for situations where the tuning of the Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) parameters fail to provide adequate prevention of the entering of the dormant state.

The operation after the recovery of a failed path also impacts the performance of the protocol. With the procedures specified in

[RFC4960] SCTP will, after a failover from the primary path, switch back to use the primary path for data transfer as soon as this path becomes available again. From a performance perspective such a forced switchback of the data transmission path can be suboptimal as the CWND towards the original primary destination address has to be rebuilt once data transfer resumes, [CARO02]. As an optional alternative to the switchback operation of [RFC4960], this document specifies an alternative Primary Path Switchover procedure which avoid such forced switchbacks of the data transfer path. The Primary Path Switchover operation was originally proposed in [CARO02].

While SCTP-PF primarily is motivated by a desire to improve the multi-homed operation, the feature applies also to SCTP single-homed operation. Here the algorithm serves to provide increased failure detection on idle associations, whereas the failover or switchback aspects of the algorithm will not be activated. This is discussed in more detail in Appendix C.

A brief description of the motivation for the introduction of the Potentially Failed state including a discussion of alternative approaches to mitigate the deficiencies of the [RFC4960] failover operation are given in the Appendices. Discussion of path bouncing effects that might be caused by frequent switchovers, are also provided there.

2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. SCTP with Potentially Failed Destination State (SCTP-PF)

3.1. Overview

To minimize the performance impact during failover, the sender should avoid transmitting data to a failed destination address as early as possible. In the [RFC4960] SCTP path management scheme, the sender stops transmitting data to a destination address only after the destination address is marked inactive. This process takes a significant amount of time as it requires the error counter of the destination address to exceed the Path.Max.Retrans (PMR) threshold. The issue cannot simply be mitigated by lowering of the PMR threshold because this may result in spurious failure detection and unnecessary prevention of the usage of a preferred primary path. Also due to the coupled tuning of the Path.Max.Retrans (PMR) and the Association.Max.Retrans (AMR) parameter values in [RFC4960], lowering

of the PMR threshold may result in lowering of the AMR threshold, which would result in decrease of the fault tolerance of SCTP.

The solution provided in this document is to extend the SCTP path management scheme of [RFC4960] by the addition of the Potentially Failed (PF) state as an intermediate state in between the active and inactive state of a destination address in the [RFC4960] path management scheme, and let the failover of data transfer away from a destination address be driven by the entering of the PF state instead of by the entering of the inactive state. Thereby SCTP may perform quick failover without negatively impacting the overall fault tolerance of [RFC4960] SCTP. At the same time, RTO-based HEARTBEAT probing is initiated towards a destination address once it enters PF state. Thereby SCTP may quickly ascertain whether network connectivity towards the destination address is broken or whether the failover was spurious. In the case where the failover was spurious data transfer may quickly resume towards the original destination address.

The new failure detection algorithm assumes that loss detected by a timeout implies either severe congestion or network connectivity failure. It recommends that by default a destination address is classified as PF at the occurrence of the first timeout.

3.2. Specification of the SCTP-PF Procedures

The SCTP-PF operation is specified as follows:

1. The sender maintains a new tunable SCTP Protocol Parameter called PotentiallyFailed.Max.Retrans (PFMR). The PFMR defines the new intermediate PF threshold on the destination address error counter. When this threshold is exceeded the destination address is classified as PF. The RECOMMENDED value of PFMR is 0. If PFMR is set to be greater than or equal to Path.Max.Retrans (PMR), the resulting PF threshold will be so high that the destination address will reach the inactive state before it can be classified as PF.

2. The error counter of an active destination address is incremented or cleared as specified in [RFC4960]. This means that the error counter of the destination address in active state will be incremented each time the T3-rtx timer expires, or each time a HEARTBEAT chunk is sent when idle and not acknowledged within an RTO. When the value in the destination address error counter exceeds PFMR, the endpoint MUST mark the destination address as in the PF state.

3. A SCTP-PF sender SHOULD NOT send data to destination addresses in PF state when alternative destination addresses in active state are available. Specifically this means that:

   i When there is outbound data to send and the destination address presently used for data transmission is in PF state, the sender SHOULD choose a destination address in active state, if one exists, and use this destination address for data transmission.

   ii As specified in [RFC4960] section 6.4.1, when the sender retransmits data that has timed out, it should attempt to pick a new destination address for data retransmission. In this case, the sender SHOULD choose an alternate destination transport address in active state if one exists.

   iii When there is outbound data to send and the SCTP user explicitly requests to send data to a destination address in PF state, the sender SHOULD send the data to an alternate destination address in active state if one exists.

   When choosing among multiple destination addresses in active state an SCTP sender will follow the guiding principles of section 6.4.1 of [RFC4960] of choosing most divergent source-destination pairs compared with, for i.: the destination address in PF state that it performs a failover from, and for ii.: the destination address towards which the data timed out. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document.

   In all cases, the sender MUST NOT change the state of chosen destination address, whether this state be active or PF, and it MUST NOT clear the error counter of the destination address as a result of choosing the destination address for data transmission.

4. When the destination addresses are all in PF state or some in PF state and some in inactive state, the sender MUST choose one destination address in PF state and SHOULD transmit or retransmit data to this destination address using the following rules:

   A. The sender SHOULD choose the destination in PF state with the lowest error count (fewest consecutive timeouts) for data transmission and transmit or retransmit data to this destination.

B. When there are multiple destination addresses in PF state with same error count, the sender should let the choice among the multiple destination addresses in PF state with equal error count be based on the [RFC4960], section 6.4.1, principles of choosing most divergent source-destination pairs when executing (potentially consecutive) retransmission. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document.

The sender MUST NOT change the state and the error counter of any destination addresses as the result of the selection.

5. The HB.interval of the Path Heartbeat function of [RFC4960] MUST be ignored for destination addresses in PF state. Instead HEARTBEAT chunks are sent to destination addresses in PF state once per RTO. HEARTBEAT chunks SHOULD be sent to destination addresses in PF state, but the sending of HEARTBEATS MUST honor whether the Path Heartbeat function (Section 8.3 of [RFC4960]) is enabled for the destination address or not. I.e., if the Path Heartbeat function is disabled for the destination address in question, HEARTBEATS MUST NOT be sent. Note that when Heartbeat function is disabled, it may take longer to transition a destination address in PF state back to active state.

6. HEARTBEATs are sent when a destination address reaches the PF state. When a HEARTBEAT chunk is not acknowledged within the RTO, the sender increments the error counter and exponentially backs off the RTO value. If the error counter is less than PMR, the sender transmits another packet containing the HEARTBEAT chunk immediately after timeout expiration on the previous HEARTBEAT. When data is being transmitted to a destination address in the PF state, the transmission of a HEARTBEAT chunk MAY be omitted in case where the receipt of a SACK of the data or a T3-rtx timer expiration on the data can provide equivalent information, such as the case where the data chunk has been transmitted to a single destination address only. Likewise, the timeout of a HEARTBEAT chunk MAY be ignored if data is outstanding towards the destination address.

7. When the sender receives a HEARTBEAT ACK from a HEARTBEAT sent to a destination address in PF state, the sender SHOULD clear the error counter of the destination address and transition the destination address back to active state. However, there may be a situation where HEARTBEAT chunks can go through while DATA chunks cannot. Hence, in a situation where a HEARTBEAT ACK arrives while there is data outstanding towards the destination address to which the HEARTBEAT was sent, then an implementation

MAY choose to not have the HEARTBEAT ACK reset the error counter, but have the error counter reset await the fate of the outstanding data transmission. This situation can happen when data is sent to a destination address in PF state. When the sender resumes data transmission on a destination address after a transition of the destination address from PF to active state, it MUST do this following the prescriptions of Section 7.2 of [RFC4960].

8. Additional (PMR - PFMR) consecutive timeouts on a destination address in PF state confirm the path failure, upon which the destination address transitions to the inactive state. As described in [RFC4960], the sender (i) SHOULD notify the ULP about this state transition, and (ii) transmit HEARTBEAT chunks to the inactive destination address at a lower HB.interval frequency as described in Section 8.3 of [RFC4960] (when the Path Heartbeat function is enabled for the destination address).

9. Acknowledgments for chunks that have been transmitted to multiple destinations (i.e., a chunk which has been retransmitted to a different destination address than the destination address to which the chunk was first transmitted) SHOULD NOT clear the error count for an inactive destination address and SHOULD NOT move a destination address in PF state back to active state, since a sender cannot disambiguate whether the ACK was for the original transmission or the retransmission(s). A SCTP sender MAY clear the error counter and move a destination address back to active state by information other than acknowledgments, when it can uniquely determine which destination, among multiple destination addresses, the chunk reached. This document makes no reference to what such information could consist of, nor how such information could be obtained.

10. Acknowledgments for data chunks that has been transmitted to one destination address only MUST clear the error counter for the destination address and MUST transition a destination address in PF state back to active state. This situation can happen when new data is sent to a destination address in the PF state. It can also happen in situations where the destination address is in the PF state due to the occurrence of a spurious T3-rtx timer and acknowledgments start to arrive for data sent prior to occurrence of the spurious T3-rtx and data has not yet been retransmitted towards other destinations. This document does not specify special handling for detection of or reaction to spurious T3-rtx timeouts, e.g., for special operation vis-a-vis the congestion control handling or data retransmission operation towards a destination address which undergoes a transition from

active to PF to active state due to a spurious T3-rtx timeout. But it is noted that this is an area which would benefit from additional attention, experimentation and specification for single-homed SCTP as well as for multi-homed SCTP protocol operation.

11. When all destination addresses are in inactive state, and SCTP protocol operation thus is said to be in dormant state, the prescriptions given in Section 4 shall be followed.

12. The SCTP stack SHOULD expose the PF state of its destination addresses to the ULP as well as provide the means to notify the ULP of state transitions of its destination addresses from active to PF, and vice-versa. However it is recommended that an SCTP stack implementing SCTP-PF also allows for that the ULP is kept ignorant of the PF state of its destinations and the associated state transitions, thus allowing for retain of the simpler state transition model of RFC4960 in the ULP. For this reason it is recommended that an SCTP stack implementing SCTP-PF also provides the ULP with the means to suppress exposure of the PF state and the associated state transitions.

4. Dormant State Operation

In a situation with complete disruption of the communication in between the SCTP Endpoints, the aggressive HEARTBEAT transmissions of SCTP-PF on destination addresses in PF state may make the association enter dormant state faster than a standard [RFC4960] SCTP implementation given the same setting of Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR). For example, an SCTP association with two destination addresses typically would reach dormant state in half the time of an [RFC4960] SCTP implementation in such situations. This is because a SCTP PF sender will send HEARTBEATS and data retransmissions in parallel with RTO intervals when there are multiple destinations addresses in PF state. This argument presumes that RTO << HB.interval of [RFC4960]. With the design goal that SCTP-PF shall provide the same level of disruption tolerance as an [RFC4960] SCTP implementation with the same Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) setting, we prescribe for that an SCTP-PF implementation SHOULD operate as described below in Section 4.1 during dormant state.

An SCTP-PF implementation MAY choose a different dormant state operation than the one described below in Section 4.1 provided that the solution chosen does not decrease the fault tolerance of the SCTP-PF operation.

The below prescription for SCTP-PF dormant state handling MUST NOT be coupled to the value of the PFMR, but solely to the activation of SCTP-PF logic in an SCTP implementation.

It is noted that the below dormant state operation is considered to provide added disruption tolerance also for an [RFC4960] SCTP implementation, and that it can be sensible for an [RFC4960] SCTP implementation to follow this mode of operation. For an [RFC4960] SCTP implementation the continuation of data transmission during dormant state makes the fault tolerance of SCTP be more robust towards situations where some, or all, alternative paths of an SCTP association approach, or reach, inactive state before the primary path used for data transmission observes trouble.

4.1. SCTP Dormant State Procedure

a. When the destination addresses are all in inactive state and data is available for transfer, the sender MUST choose one destination and transmit data to this destination address.

b. The sender MUST NOT change the state of the chosen destination address (it remains in inactive state) and it MUST NOT clear the error counter of the destination address as a result of choosing the destination address for data transmission.

c. The sender SHOULD choose the destination in inactive state with the lowest error count (fewest consecutive timeouts) for data transmission. When there are multiple destinations with same error count in inactive state, the sender SHOULD attempt to pick the most divergent source - destination pair from the last source - destination pair where failure was observed. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document. To support differentiation of inactive destination addresses based on their error count SCTP will need to allow for increment of the destination address error counters up to some reasonable limit above PMR+1, thus changing the prescriptions of [RFC4960], section 8.3, in this respect. The exact limit to apply is not specified in this document but it is considered reasonable to require for the limit to be an order of magnitude higher than the PMR value. A sender MAY choose to deploy other strategies that the strategy defined here. The strategy to prioritize the last active destination address, i.e., the destination address with the fewest error counts is optimal when some paths are permanently inactive, but suboptimal when a path instability is transient.

5. Primary Path Switchover

The objective of the Primary Path Switchover operation is to allow the SCTP sender to continue data transmission on a new working path even when the old primary destination address becomes active again. This is achieved by having SCTP perform a switchover of the primary path to the new working path if the error counter of the primary path exceeds a certain threshold. This mode of operation can be applied not only to SCTP-PF implementations, but also to [RFC4960] implementations.

The Primary Path Switchover operation requires only sender side changes. The details are:

1. The sender maintains a new tunable parameter, called Primary.Switchover.Max.Retrans (PSMR). For SCTP-PF implementations, the PSMR MUST be set greater or equal to the PFMR value. For [RFC4960] implementations the PSMR MUST be set greater or equal to the PMR value. Implementations MUST reject any other values of PSMR.

2. When the path error counter on a set primary path exceeds PSMR, the SCTP implementation MUST autonomously select and set a new primary path.

3. The primary path selected by the SCTP implementation MUST be the path which at the given time would be chosen for data transfer. A previously failed primary path can be used as data transfer path as per normal path selection when the present data transfer path fails.

4. For SCTP-PF, the recommended value of PSMR is PFMR when Primary Path Switchover operation mode is used. This means that no forced switchback to a previously failed primary path is performed. An SCTP-PF implementation of Primary Path Switchover MUST support the setting of PSMR = PFMR. A SCTP-PF implementation of Primary Path Switchover MAY support setting of PSMR > PFMR.

5. For [RFC4960] SCTP, the recommended value of PSMR is PMR when Primary Path Switchover is used. This means that no forced switchback to a previously failed primary path is performed. A [RFC4960] SCTP implementation of Primary Path Switchover MUST support the setting of PSMR = PMR. An [RFC4960] SCTP implementation of Primary Path Switchover MAY support larger settings of PSMR > PMR.

6. It MUST be possible to disable the Primary Path Switchover operation and obtain the standard switchback operation of [RFC4960].

The manner of switchover operation that is most optimal in a given scenario depends on the relative quality of a set primary path versus the quality of alternative paths available as well as on the extent to which it is desired for the mode of operation to enforce traffic distribution over a number of network paths. I.e., load distribution of traffic from multiple SCTP associations may be sought to be enforced by distribution of the set primary paths with [RFC4960] switchback operation. However as [RFC4960] switchback behavior is suboptimal in certain situations, especially in scenarios where a number of equally good paths are available, an SCTP implementation MAY support also, as alternative behavior, the Primary Path Switchover mode of operation and MAY enable it based on applications' requests.

For an SCTP implementation that implements the Primary Path Switchover operation, this specification RECOMMENDS that the standard RFC4960 switchback operation is retained as the default operation.

6. Suggested SCTP Protocol Parameter Values

This document does not alter the [RFC4960] value recommendation for the SCTP Protocol Parameters defined in [RFC4960].

The following protocol parameter is RECOMMENDED:

PotentiallyFailed.Max.Retrans (PFMR) - 0

7. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to control and observe the SCTP-PF behavior as well as the Primary Path Switchover function.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is, by means of the existing SCTP_PEER_ADDR_CHANGE event, extended to provide the event notification when a peer address enters or leaves the potentially failed state as well as the socket API implementation is extended to expose the potentially failed state of a peer address in the existing SCTP_GET_PEER_ADDR_INFO structure.

Furthermore, two new read/write socket options for the level IPPROTO_SCTP and the name SCTP_PEER_ADDR_THLDS and

SCTP_EXPOSE_POTENTIALLY_FAILED_STATE are defined as described below. The first socket option is used to control the values of the PFMR and PSMR parameters described in Section 3 and in Section 5. The second one controls the exposition of the potentially failed path state.

Support for the SCTP_PEER_ADDR_THLDS and SCTP_EXPOSE_POTENTIALLY_FAILED_STATE socket options need also to be added to the function sctp_opt_info().

7.1. Support for the Potentially Failed Path State

As defined in [RFC6458], the SCTP_PEER_ADDR_CHANGE event is provided if the status of a peer address changes. In addition to the state changes described in [RFC6458], this event is also provided, if a peer address enters or leaves the potentially failed state. The notification as defined in [RFC6458] uses the following structure:

```
struct sctp_paddr_change {
  uint16_t spc_type;
  uint16_t spc_flags;
  uint32_t spc_length;
  struct sockaddr_storage spc_aaddr;
  uint32_t spc_state;
  uint32_t spc_error;
  sctp_assoc_t spc_assoc_id;
}
```

[RFC6458] defines the constants SCTP_ADDR_AVAILABLE, SCTP_ADDR_UNREACHABLE, SCTP_ADDR_REMOVED, SCTP_ADDR_ADDED, and SCTP_ADDR_MADE_PRIM to be provided in the spc_state field. This document defines in addition to that the new constant SCTP_ADDR_POTENTIALLY_FAILED, which is reported if the affected address becomes potentially failed.

The SCTP_GET_PEER_ADDR_INFO socket option defined in [RFC6458] can be used to query the state of a peer address. It uses the following structure:

```
struct sctp_paddrinfo {
  sctp_assoc_t spinfo_assoc_id;
  struct sockaddr_storage spinfo_address;
  int32_t spinfo_state;
  uint32_t spinfo_cwnd;
  uint32_t spinfo_srtt;
  uint32_t spinfo_rto;
  uint32_t spinfo_mtu;
};
```

[RFC6458] defines the constants SCTP_UNCONFIRMED, SCTP_ACTIVE, and SCTP_INACTIVE to be provided in the spinfo_state field. This document defines in addition to that the new constant SCTP_POTENTIALLY_FAILED, which is reported if the peer address is potentially failed.

7.2. Peer Address Thresholds (SCTP_PEER_ADDR_THLDS) Socket Option

Applications can control the SCTP-PF behavior by getting or setting the number of consecutive timeouts before a peer address is considered potentially failed or unreachable. The same socket option is used by applications to set and get the number of timeouts before the primary path is changed automatically by the Primary Path Switchover function. This socket option uses the level IPPROTO_SCTP and the name SCTP_PEER_ADDR_THLDS.

The following structure is used to access and modify the thresholds:

```
struct sctp_paddrthlds {
  sctp_assoc_t spt_assoc_id;
  struct sockaddr_storage spt_address;
  uint16_t spt_pathmaxrxt;
  uint16_t spt_pathpfthld;
  uint16_t spt_pathcpthld;
};
```

spt_assoc_id: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application may fill in an association identifier or SCTP_FUTURE_ASSOC. It is an error to use SCTP_{CURRENT|ALL}_ASSOC in spt_assoc_id.

spt_address: This specifies which peer address is of interest. If a wild card address is provided, this socket option applies to all current and future peer addresses.

spt_pathmaxrxt: Each peer address of interest is considered unreachable, if its path error counter exceeds spt_pathmaxrxt.

spt_pathpfthld: Each peer address of interest is considered Potentially Failed, if its path error counter exceeds spt_pathpfthld.

spt_pathcpthld: Each peer address of interest is not considered the primary remote address anymore, if its path error counter exceeds spt_pathcpthld. Using a value of 0xffff disables the selection of a new primary peer address. If an implementation does not support the automatically selection of a new primary address, it should indicate an error with errno set to EINVAL if a value different

from 0xffff is used in spt_pathcpthld. For SCTP-PF, the setting of spt_pathcpthld < spt_pathpfthld should be rejected with errno set to EINVAL. For [RFC4960] SCTP, the setting of spt_pathcpthld < spt_pathmaxrxt should be rejected with errno set to EINVAL. A SCTP-PF implementation may support only setting of spt_pathcpthld = spt_pathpfthld and spt_pathcpthld = 0xffff and a [RFC4960] SCTP implementation may support only setting of spt_pathcpthld = spt_pathmaxrxt and spt_pathcpthld = 0xffff. In these cases SCTP shall reject setting of other values with errno set to EINVAL.

7.3. Exposing the Potentially Failed Path State (SCTP_EXPOSE_POTENTIALLY_FAILED_STATE) Socket Option

Applications can control the exposure of the potentially failed path state in the SCTP_PEER_ADDR_CHANGE event and the SCTP_GET_PEER_ADDR_INFO as described in Section 7.1. The default value is implementation specific.

This socket option uses the level IPPROTO_SCTP and the name SCTP_EXPOSE_POTENTIALLY_FAILED_STATE.

The following structure is used to control the exposition of the potentially failed path state:

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application may fill in an association identifier or SCTP_FUTURE_ASSOC. It is an error to use SCTP_{CURRENT|ALL}_ASSOC in assoc_id.

assoc_value: The potentially failed path state is exposed if and only if this parameter is non-zero.

8. Security Considerations

Security considerations for the use of SCTP and its APIs are discussed in [RFC4960] and [RFC6458].

The logic introduced by this document does not impact existing SCTP messages on the wire. Also, this document does not introduce any new SCTP messages on the wire that require new security considerations.

SCTP-PF makes SCTP not only more robust during primary path failure/ congestion but also more vulnerable to network connectivity/

congestion attacks on the primary path. SCTP-PF makes it easier for an attacker to trick SCTP to change data transfer path, since the duration of time that an attacker needs to negatively influence the network connectivity is much shorter than [RFC4960]. However, SCTP-PF does not constitute a significant change in the duration of time and effort an attacker needs to keep SCTP away from the primary path. With the standard switchback operation [RFC4960] SCTP resumes data transfer on its primary path as soon as the next HEARTBEAT succeeds.

On the other hand, usage of the Primary Path Switchover mechanism, does change the threat analysis. This is because on-path attackers can force a permanent change of the data transfer path by blocking the primary path until the switchover of the primary path is triggered by the Primary Path Switchover algorithm. This especially will be the case when the Primary Path Switchover is used together with SCTP-PF with the particular setting of PSMR = PFMR = 0, as Primary Path Switchover here happens already at the first RTO timeout experienced. Users of the Primary Path Switchover mechanism should be aware of this fact.

The event notification of path state transfer from active to potentially failed state and vice versa gives attackers an increased possibility to generate more local events. However, it is assumed that event notifications are rate-limited in the implementation to address this threat.

9. MIB Considerations

SCTP-PF introduces new SCTP algorithms for failover and switchback with associated new state parameters. It is recommended that the SCTP-MIB defined in [RFC3873] is updated to support the management of the SCTP-PF implementation. This can be done by extending the sctpAssocRemAddrActive field of the SCTPAssocRemAddrTable to include information of the PF state of the destination address and by adding new fields to the SCTPAssocRemAddrTable supporting PotentiallyFailed.Max.Retrans (PFMR) and Primary.Switchover.Max.Retrans (PSMR) parameters.

10. IANA Considerations

This document does not create any new registries or modify the rules for any existing registries managed by IANA.

11. Acknowledgements

The authors wish to thank Michael Tuexen for his many invaluable comments and for his very substantial support with the making of this document.

12. Proposed Change of Status (to be Deleted before Publication)

Initially this work looked to entail some changes of the Congestion Control (CC) operation of SCTP and for this reason the work was proposed as Experimental. These intended changes of the CC operation have since been judged to be irrelevant and are no longer part of the specification. As the specification entails no other potential harmful features, consensus exists in the WG to bring the work forward as PS.

Initially concerns have been expressed about the possibility for the mechanism to introduce path bouncing with potential harmful network impacts. These concerns are believed to be unfounded. This issue is addressed in Appendix B.

It is noted that the feature specified by this document is implemented by multiple SCTP SW implementations and furthermore that various variants of the solution have been deployed in telephony signaling environments for several years with good results.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

13.2. Informative References

[CARO02] Caro Jr., A., Iyengar, J., Amer, P., Heinz, G., and R. Stewart, "A Two-level Threshold Recovery Mechanism for SCTP", Tech report, CIS Dept, University of Delaware , 7 2002.

[CARO04] Caro Jr., A., Amer, P., and R. Stewart, "End-to-End Failover Thresholds for Transport Layer Multi homing", MILCOM 2004 , 11 2004.

[CARO05] Caro Jr., A., "End-to-End Fault Tolerance using Transport Layer Multi homing", Ph.D Thesis, University of Delaware , 1 2005.

[FALLON08]
Fallon, S., Jacob, P., Qiao, Y., Murphy, L., Fallon, E., and A. Hanley, "SCTP Switchover Performance Issues in WLAN Environments", IEEE CCNC 2008, 1 2008.

[GRINNEMO04]
Grinnemo, K-J. and A. Brunstrom, "Performance of SCTP-controlled failovers in M3UA-based SIGTRAN networks", Advanced Simulation Technologies Conference , 4 2004.

[IYENGAR06]
Iyengar, J., Amer, P., and R. Stewart, "Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-end Paths.", IEEE/ACM Trans on Networking 14(5), 10 2006.

[JUNGMAIER02]
Jungmaier, A., Rathgeb, E., and M. Tuexen, "On the use of SCTP in failover scenarios", World Multiconference on Systemics, Cybernetics and Informatics , 7 2002.

[NATARAJAN09]
Natarajan, P., Ekiz, N., Amer, P., and R. Stewart, "Concurrent Multipath Transfer during Path Failure", Computer Communications , 5 2009.

[RFC3873] Pastor, J. and M. Belinchon, "Stream Control Transmission Protocol (SCTP) Management Information Base (MIB)", RFC 3873, DOI 10.17487/RFC3873, September 2004, <http://www.rfc-editor.org/info/rfc3873>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, December 2011.

Appendix A. Discussions of Alternative Approaches

This section lists alternative approaches for the issues described in this document. Although these approaches do not require to update RFC4960, we do not recommend them from the reasons described below.

A.1. Reduce Path.Max.Retrans (PMR)

Smaller values for Path.Max.Retrans shorten the failover duration and in fact this is recommended in some research results [JUNGMAIER02] [GRINNEMO04] [FALLON08]. However to significantly reduce the failover time it is required to go down (as with PFMR) to Path.Max.Retrans=0 and with this setting SCTP switches to another

destination address already on a single timeout which may result in spurious failover. Spurious failover is a problem in [RFC4960] SCTP as the transmission of HEARTBEATS on the left primary path, unlike in SCTP-PF, is governed by 'HB.interval' also during the failover process. 'HB.interval' is usually set in the order of seconds (recommended value is 30 seconds) and when the primary path becomes inactive, the next HEARTBEAT may be transmitted only many seconds later. Indeed as recommended, only 30 secs later. Meanwhile, the primary path may since long have recovered, if it needed recovery at all (indeed the failover could be truly spurious). In such situations, post failover, an endpoint is forced to wait in the order of many seconds before the endpoint can resume transmission on the primary path and furthermore once it returns on the primary path the CWND needs to be rebuild anew - a process which the throughput already have had to suffer from on the alternate path. Using a smaller value for 'HB.interval' might help this situation, but it would result in a general waste of bandwidth as such more frequent HEARTBEATING would take place also when there are no observed troubles. The bandwidth overhead may be diminished by having the ULP use a smaller 'HB.interval' only on the path which at any given time is set to be the primary path, but this adds complication in the ULP.

In addition, smaller Path.Max.Retrans values also affect the 'Association.Max.Retrans' value. When the SCTP association's error count exceeds Association.Max.Retrans threshold, the SCTP sender considers the peer endpoint unreachable and terminates the association. Section 8.2 in [RFC4960] recommends that Association.Max.Retrans value should not be larger than the summation of the Path.Max.Retrans of each of the destination addresses. Else the SCTP sender considers its peer reachable even when all destinations are INACTIVE and to avoid this dormant state operation, [RFC4960] SCTP implementation SHOULD reduce Association.Max.Retrans accordingly whenever it reduces Path.Max.Retrans. However, smaller Association.Max.Retrans value decreases the fault tolerance of SCTP as it increases the chances of association termination during minor congestion events.

A.2. Adjust RTO related parameters

As several research results indicate, we can also shorten the duration of failover process by adjusting RTO related parameters [JUNGMAIER02] [FALLON08]. During failover process, RTO keeps being doubled. However, if we can choose smaller value for RTO.max, we can stop the exponential growth of RTO at some point. Also, choosing smaller values for RTO.initial or RTO.min can contribute to keep the RTO value small.

Similar to reducing Path.Max.Retrans, the advantage of this approach is that it requires no modification to the current specification, although it needs to ignore several recommendations described in the Section 15 of [RFC4960]. However, this approach requires to have enough knowledge about the network characteristics between end points. Otherwise, it can introduce adverse side-effects such as spurious timeouts.

The significant issue with this approach, however, is that even if the RTO.max is lowered to an optimal low value, then as long as the Path.Max.Retrans is kept at the [RFC4960] recommended value, the reduction of the RTO.max doesn’t reduce the failover time sufficiently enough to prevent severe performance degradation during failover.

Appendix B. Discussions for Path Bouncing Effect

The methods described in the document can accelerate the failover process. Hence, they might introduce the path bouncing effect where the sender keeps changing the data transmission path frequently. This sounds harmful to the data transfer, however several research results indicate that there is no serious problem with SCTP in terms of path bouncing effect [CARO04] [CARO05].

There are two main reasons for this. First, SCTP is basically designed for multipath communication, which means SCTP maintains all path related parameters (CWND, ssthresh, RTT, error count, etc) per each destination address. These parameters cannot be affected by path bouncing. In addition, when SCTP migrates the data transfer to another path, it starts with the minimal or the initial CWND. Hence, there is little chance for packet reordering or duplicating.

Second, even if all communication paths between the end-nodes share the same bottleneck, the SCTP-PF results in a behavior already allowed by [RFC4960].

Appendix C. SCTP-PF for SCTP Single-homed Operation

For a single-homed SCTP association the only tangible effect of the activation of SCTP-PF operation is enhanced failure detection in terms of potential notification of the PF state of the sole destination address as well as, for idle associations, more rapid entering, and notification, of inactive state of the destination address and more rapid end-point failure detection. It is believed that neither of these effects are harmful, provided adequate dormant state operation is implemented, and furthermore that they may be particularly useful for applications that deploys multiple SCTP associations for load balancing purposes. The early notification of

the PF state may be used for preventive measures as the entering of the PF state can be used as a warning of potential congestion. Depending on the PMR value, the aggressive HEARTBEAT transmission in PF state may speed up the end-point failure detection (exceed of AMR threshold on the sole path error counter) on idle associations in case where relatively large HB.interval value compared to RTO (e.g. 30secs) is used.

Authors' Addresses

Yoshifumi Nishida
GE Global Research
2623 Camino Ramon
San Ramon, CA 94583
USA

Email: nishida@wide.ad.jp

Preethi Natarajan
Cisco Systems
510 McCarthy Blvd
Milpitas, CA 95035
USA

Email: prenatar@cisco.com

Armando Caro
BBN Technologies
10 Moulton St.
Cambridge, MA 02138
USA

Email: acaro@bbn.com

Paul D. Amer
University of Delaware
Computer Science Department - 434 Smith Hall
Newark, DE 19716-2586
USA

Email: amer@udel.edu

Karen E. E. Nielsen
Ericsson
Kistavaegen 25
Stockholm 164 80
Sweden

Email: karen.nielsen@tieto.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 5, 2018

R. Stewart
Netflix, Inc.
M. Tuexen
Muenster Univ. of Appl. Sciences
S. Loreto
Ericsson
R. Seggelmann
Metafinanz Informationssysteme GmbH

September 1, 2017

Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol

draft-ietf-tsvwg-sctp-ndata-13.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a message oriented transport protocol supporting arbitrarily large user messages. This document adds a new chunk to SCTP for carrying payload data. This allows a sender to interleave different user messages that would otherwise result in head of line blocking at the sender. The interleaving of user messages is required for WebRTC Datachannels.

Whenever an SCTP sender is allowed to send user data, it may choose from multiple outgoing SCTP streams. Multiple ways for performing this selection, called stream schedulers, are defined in this document. A stream scheduler can choose to either implement, or not implement, user message interleaving.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Overview

When SCTP [RFC4960] was initially designed it was mainly envisioned for the transport of small signaling messages. Late in the design stage it was decided to add support for fragmentation and reassembly of larger messages with the thought that someday Session Initiation Protocol (SIP) [RFC3261] style signaling messages may also need to use SCTP and a single Maximum Transmission Unit (MTU) sized message would be too small. Unfortunately this design decision, though valid at the time, did not account for other applications that might send large messages over SCTP. The sending of such large messages over SCTP as specified in [RFC4960] can result in a form of sender side head of line blocking (e.g., when the transmission of a message is blocked from transmission because the sender has started the transmission of another, possibly large, message). This head of line blocking is caused by the use of the Transmission Sequence Number (TSN) for three different purposes:

1. As an identifier for DATA chunks to provide a reliable transfer.

2. As an identifier for the sequence of fragments to allow reassembly.

3. As a sequence number allowing up to 2**16 - 1 Stream Sequence Numbers (SSNs) outstanding.

The protocol requires all fragments of a user message to have consecutive TSNs. This document allows an SCTP sender to interleave different user messages.

This document also defines several stream schedulers for general SCTP associations allowing different relative stream treatments. The stream schedulers may behave differently depending on whether user message interleaving has been negotiated for the association or not.

Figure 1 illustrates the behaviour of a round robin stream scheduler using DATA chunks when three streams with the Stream Identifiers

(SIDs) 0, 1, and 2 are used. Each queue for SID 0 and SID 2 contains a single user message requiring three chunks, the queue for SID 1 contains three user messages each requiring a single chunk. It is shown how these user messages are encapsulated in chunk using TSN 0 to TSN 8. Please note that the use of such a scheduler implies late TSN assignment but it can be used with an [RFC4960] compliant implementation that does not support user message interleaving. Late TSN assignment means that the sender generates chunks from user messages and assigns the TSN as late as possible in the process of sending the user messages.

```
+---+---+---+
|    0/0    |-+
+---+---+---+ |
              |  +---+---+---+---+---+---+---+---+---+
+---+---+---+ +->|1/2|1/1|2/0|2/0|2/0|1/0|0/0|0/0|0/0|
|1/2|1/1|1/0|--->|---|---|---|---|---|---|---|---|---|
+---+---+---+ +->| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
              |  +---+---+---+---+---+---+---+---+---+
+---+---+---+ |
|    2/0    |-+
+---+---+---+
                                +-------+
  +-------+                     |SID/SSN|
  |SID/SSN|                     |-------|
  +-------+                     |  TSN  |
                                +-------+
```

Figure 1: Round Robin Scheduler without User Message Interleaving

This document describes a new chunk carrying payload data called I-DATA. This chunk incorporates the properties of the current SCTP DATA chunk, all the flags and fields except the Stream Sequence Number (SSN), but also adds two new fields in its chunk header, the Fragment Sequence Number (FSN) and the Message Identifier (MID). The FSN is only used for reassembling all fragments having the same MID and ordering property. The TSN is only used for the reliable transfer in combination with Selective Acknowledgment (SACK) chunks.

In addition, the MID is also used for ensuring ordered delivery instead of using the stream sequence number (The I-DATA chunk omits a SSN.).

Figure 2 illustrates the behaviour of an interleaving round robin stream scheduler using I-DATA chunks.

```
+---+---+---+
|    0/0    |-+
+---+---+---+ |
              |  +-----+-----+-----+-----+-----+-----+-----+-----+-----+
+---+---+---+ +->|2/0/2|1/2/0|0/0/2|2/0/1|1/1/0|0/0/1|2/0/0|1/0/0|0/0/0|
|1/2|1/1|1/0|--->|-----|-----|-----|-----|-----|-----|-----|-----|-----|
+---+---+---+ +->|  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
              |  +-----+-----+-----+-----+-----+-----+-----+-----+-----+
+---+---+---+ |
|    2/0    |-+
+---+---+---+
                                      +-----------+
  +-------+                           |SID/MID/FSN|
  |SID/MID|                           |-----------|
  +-------+                           |    TSN    |
                                      +-----------+
```

Figure 2: Round Robin Scheduler with User Message Interleaving

The support of the I-DATA chunk is negotiated during the association setup using the Supported Extensions Parameter as defined in [RFC5061]. If I-DATA support has been negotiated for an association, I-DATA chunks are used for all user-messages. DATA chunks are not permitted when I-DATA support has been negotiated. It should be noted that an SCTP implementation supporting I-DATA chunks needs to allow the coexistence of associations using DATA chunks and associations using I-DATA chunks.

In Section 2 this document specifies the user message interleaving by defining the I-DATA chunk, the procedures to use it and its interactions with other SCTP extensions. Multiple stream schedulers are defined in Section 3 followed in Section 4 by describing an extension to the socket API for using what is specified in this document.

1.2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. User Message Interleaving

The protocol mechanisms described in this document allow the interleaving of user messages sent on different streams. They do not support the interleaving of multiple messages (ordered or unordered) sent on the same stream.

The interleaving of user messages is required for WebRTC Datachannels as specified in [I-D.ietf-rtcweb-data-channel].

An SCTP implementation supporting user message interleaving is REQUIRED to support the coexistence of associations using DATA chunks and associations using I-DATA chunks. If an SCTP implementation supports user message interleaving and the Partial Reliability extension described in [RFC3758] or the Stream Reconfiguration Extension described in [RFC6525], it is REQUIRED to implement the corresponding changes specified in Section 2.3.

2.1. The I-DATA Chunk Supporting User Message Interleaving

The following Figure 3 shows the new I-DATA chunk allowing user message interleaving.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 64   |  Res  |I|U|B|E|       Length = Variable       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              TSN                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Stream Identifier      |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Message Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Payload Protocol Identifier / Fragment Sequence Number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                           User Data                           /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: I-DATA chunk format

The only differences between the I-DATA chunk in Figure 3 and the DATA chunk defined in [RFC4960] and [RFC7053] are the addition of the new Message Identifier (MID) and the new Fragment Sequence Number (FSN) and the removal of the Stream Sequence Number (SSN). The Payload Protocol Identifier (PPID) already defined for DATA chunks in [RFC4960] and the new FSN are stored at the same location of the packet using the B bit to determine which value is stored at the location. The length of the I-DATA chunk header is 20 bytes, which is 4 bytes more than the length of the DATA chunk header defined in [RFC4960] and [RFC7053].

The old fields are:

Res: 4 bits
These bits are reserved. They MUST be set to 0 by the sender and MUST be ignored by the receiver.

I bit: 1 bit
The (I)mmediate Bit, if set, indicates that the receiver SHOULD NOT delay the sending of the corresponding SACK chunk. Same as the I bit for DATA chunks as specified in [RFC7053].

U bit: 1 bit
The (U)nordered bit, if set, indicates the user message is unordered. Same as the U bit for DATA chunks as specified in [RFC4960].

B bit: 1 bit
The (B)eginning fragment bit, if set, indicates the first fragment of a user message. Same as the B bit for DATA chunks as specified in [RFC4960].

E bit: 1 bit
The (E)nding fragment bit, if set, indicates the last fragment of a user message. Same as the E bit for DATA chunks as specified in [RFC4960].

Length: 16 bits (unsigned integer)
This field indicates the length of the DATA chunk in bytes from the beginning of the type field to the end of the User Data field excluding any padding. Similar to the Length for DATA chunks as specified in [RFC4960].

TSN: 32 bits (unsigned integer)
This value represents the TSN for this I-DATA chunk. Same as the TSN for DATA chunks as specified in [RFC4960].

Stream Identifier: 16 bits (unsigned integer)
Identifies the stream to which the user data belongs. Same as the Stream Identifier for DATA chunks as specified in [RFC4960].

The new fields are:

Reserved: 16 bits (unsigned integer)
This field is reserved. It MUST be set to 0 by the sender and MUST be ignored by the receiver.

Message Identifier (MID): 32 bits (unsigned integer)
The MID is the same for all fragments of a user message, it is used to determine which fragments (enumerated by the FSN) belong to the same user message. For ordered user messages, the MID is

also used by the SCTP receiver to deliver the user messages in the correct order to the upper layer (similar to the SSN of the DATA chunk defined in [RFC4960]). The sender uses for each outgoing stream two counters, one for ordered messages, one for unordered messages. All of these counters are independent and initially 0. They are incremented by 1 for each user message. Please note that the serial number arithmetic defined in [RFC1982] using SERIAL_BITS = 32 applies. Therefore, the sender MUST NOT have more than 2**31 - 1 ordered messages for each outgoing stream in flight and MUST NOT have more than 2**31 - 1 unordered messages for each outgoing stream in flight. A message is considered in flight, if at least on of its I-DATA chunks is not acknowledged in a non-renegable way (i.e. not acknowledged by the cumulative TSN Ack). Please note that the MID is in "network byte order", a.k.a. Big Endian.

Payload Protocol Identifier (PPID) / Fragment Sequence Number (FSN): 32 bits (unsigned integer)
If the B bit is set, this field contains the PPID of the user message. Note that in this case, this field is not touched by an SCTP implementation; therefore, its byte order is not necessarily in network byte order. The upper layer is responsible for any byte order conversions to this field, similar to the PPID of DATA chunks. In this case the FSN is implicitly considered to be 0. If the B bit is not set, this field contains the FSN. The FSN is used to enumerate all fragments of a single user message, starting from 0 and incremented by 1. The last fragment of a message MUST have the E bit set. Note that the FSN MAY wrap completely multiple times allowing arbitrarily large user messages. For the FSN the serial number arithmetic defined in [RFC1982] applies with SERIAL_BITS = 32. Therefore, a sender MUST NOT have more than 2**31 - 1 fragments of a single user message in flight. A fragment is considered in flight, if it is not acknowledged in a non-renegable way. Please note that the FSN is in "network byte order", a.k.a. Big Endian.

2.2. Procedures

This subsection describes how the support of the I-DATA chunk is negotiated and how the I-DATA chunk is used by the sender and receiver.

The handling of the I bit for the I-DATA chunk corresponds to the handling of the I bit for the DATA chunk described in [RFC7053].

2.2.1. Negotiation

An SCTP end point indicates user message interleaving support by listing the I-DATA Chunk within the Supported Extensions Parameter as defined in [RFC5061]. User message interleaving has been negotiated for an association if both end points have indicated I-DATA support.

If user message interleaving support has been negotiated for an association, I-DATA chunks MUST be used for all user messages and DATA-chunks MUST NOT be used. If user message interleaving support has not been negotiated for an association, DATA chunks MUST be used for all user messages and I-DATA chunks MUST NOT be used.

An end point implementing the socket API specified in [RFC6458] MUST NOT indicate user message interleaving support unless the user has requested its use (e.g. via the socket API, see Section 4.3). This constraint is made since the usage of this chunk requires that the application is capable of handling interleaved messages upon reception within an association. This is not the default choice within the socket API (see the SCTP_FRAGMENT_INTERLEAVE socket option in Section 8.1.20 of [RFC6458]) thus the user MUST indicate to the SCTP implementation its support for receiving completely interleaved messages.

Note that stacks that do not implement [RFC6458] may use other methods to indicate interleaved message support and thus indicate the support of user message interleaving. The crucial point is that the SCTP stack MUST know that the application can handle interleaved messages before indicating the I-DATA support.

2.2.2. Sender Side Considerations

The sender side usage of the I-DATA chunk is quite simple. Instead of using the TSN for fragmentation purposes, the sender uses the new FSN field to indicate which fragment number is being sent. The first fragment MUST have the B bit set. The last fragment MUST have the E bit set. All other fragments MUST NOT have the B bit or E bit set. All other properties of the existing SCTP DATA chunk also apply to the I-DATA chunk, i.e. congestion control as well as receiver window conditions MUST be observed as defined in [RFC4960].

Note that the usage of this chunk implies the late assignment of the actual TSN to any chunk being sent. Each I-DATA chunk uses a single TSN. This way messages from other streams may be interleaved with the fragmented message. Please note that this is the only form of interleaving support. For example, it is not possible to interleave multiple ordered or unordered user messages from the same stream.

The sender MUST NOT process (move user data into I-DATA chunks and assign a TSN to it) more than one user message in any given stream at any time. At any time, a sender MAY process multiple user messages, each of them on different streams.

The sender MUST assign TSNs to I-DATA chunks in a way that the receiver can make progress. One way to achieve this is to assign a higher TSN to the later fragments of a user message and send out the I-DATA chunks such that the TSNs are in sequence.

2.2.3. Receiver Side Considerations

Upon reception of an SCTP packet containing an I-DATA chunk whose user message needs to be reassembled, the receiver MUST first use the SID to identify the stream, consider the U bit to determine if it is part of an ordered or unordered message, find the user message identified by the MID and finally use the FSN for reassembly of the message and not the TSN. The receiver MUST NOT make any assumption about the TSN assignments of the sender. Note that a non-fragmented message is indicated by the fact that both the E and B bits are set. A message (either ordered or unordered) may be identified as being fragmented whose E and B bits are not both set.

If I-DATA support has been negotiated for an association, the reception of a DATA chunk is a violation of the above rules and therefore the receiver of the DATA chunk MUST abort the association by sending an ABORT chunk. The ABORT chunk MAY include the ’Protocol Violation’ error cause. The same applies if I-DATA support has not been negotiated for an association and an I-DATA chunk is received.

2.3. Interaction with other SCTP Extensions

The usage of the I-DATA chunk might interfere with other SCTP extensions. Future SCTP extensions MUST describe if and how they interfere with the usage of I-DATA chunks. For the SCTP extensions already defined when this document was published, the details are given in the following subsections.

2.3.1. SCTP Partial Reliability Extension

When the SCTP extension defined in [RFC3758] is used in combination with the user message interleaving extension, the new I-FORWARD-TSN chunk MUST be used instead of the FORWARD-TSN chunk. The difference between the FORWARD-TSN and the I-FORWARD-TSN chunk is that the 16-bit Stream Sequence Number (SSN) has been replaced by the 32-bit Message Identifier (MID) and the largest skipped MID can also be provided for unordered messages. Therefore, the principle applied to

ordered message when using FORWARD-TSN chunks is applied to ordered and unordered messages when using I-FORWARD-TSN chunks.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 194  | Flags = 0x00  |      Length = Variable        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       New Cumulative TSN                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Stream Identifier       |          Reserved           |U|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Message Identifier                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Stream Identifier       |          Reserved           |U|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Message Identifier                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: I-FORWARD-TSN chunk format

The old fields are:

Flags: 8-bits (unsigned integer)
These bits are reserved. They MUST be set to 0 by the sender and MUST be ignored by the receiver. Same as the Flags for FORWARD TSN chunks as specified in [RFC3758].

Length: 16-bits (unsigned integer)
This field holds the length of the chunk. Similar to the Length for FORWARD TSN chunks as specified in [RFC3758].

New Cumulative TSN: 32-bits (unsigned integer)
This indicates the new cumulative TSN to the data receiver. Same as the New Cumulative TSN for FORWARD TSN chunks as specified in [RFC3758].

The new fields are:

Stream Identifier (SID): 16-bits (unsigned integer)
This field holds the stream number this entry refers to.

Reserved: 15 bits
This field is reserved. It MUST be set to 0 by the sender and MUST be ignored by the receiver.

U bit: 1 bit
The U bit specifies if the Message Identifier of this entry refers to unordered messages (U bit is set) or ordered messages (U bit is not set).

Message Identifier (MID): 32 bits (unsigned integer)
This field holds the largest Message Identifier for ordered or unordered messages indicated by the U bit that was skipped for the stream specified by the Stream Identifier. For ordered messages this is similar to the FORWARD-TSN chunk, just replacing the 16-bit SSN by the 32-bit MID.

Support for the I-FORWARD-TSN chunk is negotiated during the SCTP association setup via the Supported Extensions Parameter as defined in [RFC5061]. Only if both end points indicated their support of user message interleaving and the I-FORWARD-TSN chunk, the partial reliability extension is negotiated and can be used in combination with user message interleaving.

The FORWARD-TSN chunk MUST be used in combination with the DATA chunk and MUST NOT be used in combination with the I-DATA chunk. The I-FORWARD-TSN chunk MUST be used in combination with the I-DATA chunk and MUST NOT be used in combination with the DATA chunk.

If I-FORWARD-TSN support has been negotiated for an association, the reception of a FORWARD-TSN chunk is a violation of the above rules and therefore the receiver of the FORWARD-TSN chunk MUST abort the association by sending an ABORT chunk. The ABORT chunk MAY include the ’Protocol Violation’ error cause. The same applies if I-FORWARD-TSN support has not been negotiated for an association and a FORWARD-TSN chunk is received.

2.3.2. SCTP Stream Reconfiguration Extension

When an association resets the SSN using the SCTP extension defined in [RFC6525], the two counters (one for the ordered messages, one for the unordered messages) used for the MIDs MUST be reset to 0.

Since most schedulers, especially all schedulers supporting user message interleaving, require late TSN assignment, it should be noted that the implementation of [RFC6525] needs to handle this.

3. Stream Schedulers

This section defines several stream schedulers. The stream schedulers may behave differently depending on whether user message interleaving has been negotiated for the association or not. An implementation MAY implement any subset of them. If the

implementation is used for WebRTC Datachannels as specified in [I-D.ietf-rtcweb-data-channel] it MUST implement the Weighted Fair Queueing Scheduler defined in Section 3.6.

The selection of the stream scheduler is done at the sender side. There is no mechanism provided for signalling the stream scheduler being used to the receiver side or even let the receiver side influence the selection of the stream scheduler used at the sender side.

3.1. First Come First Served Scheduler (SCTP_SS_FCFS)

The simple first-come, first-served scheduler of user messages is used. It just passes through the messages in the order in which they have been delivered by the application. No modification of the order is done at all. The usage of user message interleaving does not affect the sending of the chunks, except that I-DATA chunks are used instead of DATA chunks.

3.2. Round Robin Scheduler (SCTP_SS_RR)

When not using user message interleaving, this scheduler provides a fair scheduling based on the number of user messages by cycling around non-empty stream queues. When using user message interleaving, this scheduler provides a fair scheduling based on the number of I-DATA chunks by cycling around non-empty stream queues.

3.3. Round Robin Scheduler per Packet (SCTP_SS_RR_PKT)

This is a round-robin scheduler, which only switches streams when starting to fill a new packet. It bundles only DATA or I-DATA chunks referring to the same stream in a packet. This scheduler minimizes head-of-line blocking when a packet is lost because only a single stream is affected.

3.4. Priority Based Scheduler (SCTP_SS_PRIO)

Scheduling of user messages with strict priorities is used. The priority is configurable per outgoing SCTP stream. Streams having a higher priority will be scheduled first and when multiple streams have the same priority, the scheduling between them is implementation dependent. When using user message interleaving, the sending of large lower priority user messages will not delay the sending of higher priority user messages.

3.5. Fair Capacity Scheduler (SCTP_SS_FC)

A fair capacity distribution between the streams is used. This scheduler considers the lengths of the messages of each stream and schedules them in a specific way to maintain an equal capacity for all streams. The details are implementation dependent. Using user message interleaving allows for a better realization of the fair capacity usage.

3.6. Weighted Fair Queueing Scheduler (SCTP_SS_WFQ)

A weighted fair queueing scheduler between the streams is used. The weight is configurable per outgoing SCTP stream. This scheduler considers the lengths of the messages of each stream and schedules them in a specific way to use the capacity according to the given weights. If the weight of stream S1 is n times the weight of stream S2, the scheduler should assign to stream S1 n times the capacity it assigns to stream S2. The details are implementation dependent. Using user message interleaving allows for a better realization of the capacity usage according to the given weights.

This scheduler in combination with user message interleaving is used for WebRTC Datachannels as specified in [I-D.ietf-rtcweb-data-channel].

4. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to allow applications to use the extension described in this document.

Please note that this section is informational only.

4.1. Exposure of the Stream Sequence Number (SSN)

The socket API defined in [RFC6458] defines several structures in which the SSN of a received user message is exposed to the application. The list of these structures includes:

struct sctp_sndrcvinfo
: Specified in Section 5.3.2 SCTP Header Information Structure (SCTP_SNDRCV) of [RFC6458] and marked as deprecated.

struct sctp_extrcvinfo
: Specified in Section 5.3.3 Extended SCTP Header Information Structure (SCTP_EXTRCV)of [RFC6458] and marked as deprecated.

struct sctp_rcvinfo

Specified in Section 5.3.5 SCTP Receive Information Structure (SCTP_RCVINFO) of [RFC6458].

If user message interleaving is used, the lower order 16 bits of the MID are used as the SSN when filling out these structures.

4.2. SCTP_ASSOC_CHANGE Notification

When an SCTP_ASSOC_CHANGE notification (specified in Section 6.1.1 of [RFC6458]) is delivered indicating a sac_state of SCTP_COMM_UP or SCTP_RESTART for an SCTP association where both peers support the I-DATA chunk, SCTP_ASSOC_SUPPORTS_INTERLEAVING should be listed in the sac_info field.

4.3. Socket Options

option name	data type	get	set
SCTP_INTERLEAVING_SUPPORTED	struct sctp_assoc_value	X	X
SCTP_STREAM_SCHEDULER	struct sctp_assoc_value	X	X
SCTP_STREAM_SCHEDULER_VALUE	struct sctp_stream_value	X	X

4.3.1. Enable or Disable the Support of User Message Interleaving (SCTP_INTERLEAVING_SUPPORTED)

This socket option allows the enabling or disabling of the negotiation of user message interleaving support for future associations. For existing associations it allows to query whether user message interleaving support was negotiated or not on a particular association.

This socket option uses IPPROTO_SCTP as its level and SCTP_INTERLEAVING_SUPPORTED as its name. It can be used with getsockopt() and setsockopt(). The socket option value uses the following structure defined in [RFC6458]:

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets, this parameter indicates upon which association the user is performing an action. The special

sctp_assoc_t SCTP_FUTURE_ASSOC can also be used, it is an error to use SCTP_{CURRENT|ALL}_ASSOC in assoc_id.

assoc_value: A non-zero value encodes the enabling of user message interleaving whereas a value of 0 encodes the disabling of user message interleaving.

sctp_opt_info() needs to be extended to support SCTP_INTERLEAVING_SUPPORTED.

An application using user message interleaving should also set the fragment interleave level to 2 by using the SCTP_FRAGMENT_INTERLEAVE socket option specified in Section 8.1.20 of [RFC6458]. This allows the interleaving of user messages from different streams. Please note that it does not allow the interleaving of user messages (ordered or unordered) on the same stream. Failure to set this option can possibly lead to application deadlock. Some implementations might therefore put some restrictions on setting combinations of these values. Setting the interleaving level to at least 2 before enabling the negotiation of user message interleaving should work on all platforms. Since the default fragment interleave level is not 2, user message interleaving is disabled per default.

4.3.2. Get or Set the Stream Scheduler (SCTP_STREAM_SCHEDULER)

A stream scheduler can be selected with the SCTP_STREAM_SCHEDULER option for setsockopt(). The struct sctp_assoc_value is used to specify the association for which the scheduler should be changed and the value of the desired algorithm.

The definition of struct sctp_assoc_value is the same as in [RFC6458]:

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id: Holds the identifier for the association of which the scheduler should be changed. The special SCTP_{FUTURE|CURRENT|ALL}_ASSOC can also be used. This parameter is ignored for one-to-one style sockets.

assoc_value: This specifies which scheduler is used. The following constants can be used:

SCTP_SS_DEFAULT: The default scheduler used by the SCTP implementation. Typical values are SCTP_SS_FCFS or SCTP_SS_RR.

SCTP_SS_FCFS: Use the scheduler specified in Section 3.1.

SCTP_SS_RR: Use the scheduler specified in Section 3.2.

SCTP_SS_RR_PKT: Use the scheduler specified in Section 3.3.

SCTP_SS_PRIO: Use the scheduler specified in Section 3.4. The priority can be assigned with the sctp_stream_value struct. The higher the assigned value, the lower the priority, that is the default value 0 is the highest priority and therefore the default scheduling will be used if no priorities have been assigned.

SCTP_SS_FB: Use the scheduler specified in Section 3.5.

SCTP_SS_WFQ: Use the scheduler specified in Section 3.6. The weight can be assigned with the sctp_stream_value struct.

sctp_opt_info() needs to be extended to support SCTP_STREAM_SCHEDULER.

4.3.3. Get or Set the Stream Scheduler Parameter (SCTP_STREAM_SCHEDULER_VALUE)

Some schedulers require additional information to be set for individual streams as shown in the following table:

name	per stream info
SCTP_SS_DEFAULT	n/a
SCTP_SS_FCFS	no
SCTP_SS_RR	no
SCTP_SS_RR_PKT	no
SCTP_SS_PRIO	yes
SCTP_SS_FB	no
SCTP_SS_WFQ	yes

This is achieved with the SCTP_STREAM_SCHEDULER_VALUE option and the corresponding struct sctp_stream_value. The definition of struct sctp_stream_value is as follows:

```
struct sctp_stream_value {
  sctp_assoc_t assoc_id;
  uint16_t stream_id;
  uint16_t stream_value;
};
```

assoc_id: Holds the identifier for the association of which the scheduler should be changed. The special SCTP_{FUTURE|CURRENT|ALL}_ASSOC can also be used. This parameter is ignored for one-to-one style sockets.

stream_id: Holds the stream id of the stream for which additional information has to be provided.

stream_value: The meaning of this field depends on the scheduler specified. It is ignored when the scheduler does not need additional information.

sctp_opt_info() needs to be extended to support SCTP_STREAM_SCHEDULER_VALUE.

4.4. Explicit EOR Marking

Using explicit End of Record (EOR) marking for an SCTP association supporting user message interleaving allows the user to interleave the sending of user messages on different streams.

5. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

[NOTE to RFC-Editor:

The suggested values for the chunk types and the chunk flags are tentative and to be confirmed by IANA.

]

This document (RFCXXXX) is the reference for all registrations described in this section.

Two new chunk types have to be assigned by IANA.

5.1. I-DATA Chunk

IANA should assign the chunk type for this chunk from the pool of chunks with the upper two bits set to ’01’. This requires an additional line in the "Chunk Types" registry for SCTP:

ID Value	Chunk Type	Reference
64	Payload Data supporting Interleaving (I-DATA)	[RFCXXXX]

The registration table as defined in [RFC6096] for the chunk flags of this chunk type is initially given by the following table:

Chunk Flag Value	Chunk Flag Name	Reference
0x01	E bit	[RFCXXXX]
0x02	B bit	[RFCXXXX]
0x04	U bit	[RFCXXXX]
0x08	I bit	[RFCXXXX]
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

5.2. I-FORWARD-TSN Chunk

IANA should assign the chunk type for this chunk from the pool of chunks with the upper two bits set to ’11’. This requires an additional line in the "Chunk Types" registry for SCTP:

ID Value	Chunk Type	Reference
194	I-FORWARD-TSN	[RFCXXXX]

The registration table as defined in [RFC6096] for the chunk flags of this chunk type is initially empty.

6. Security Considerations

This document does not add any additional security considerations in addition to the ones given in [RFC4960] and [RFC6458].

It should be noted that the application has to consent that it is willing to do the more complex reassembly support required for user message interleaving. When doing so, an application has to provide a reassembly buffer for each incoming stream. It has to protect itself against these buffers taking too many resources. If user message

interleaving is not used, only a single reassembly buffer needs to be provided for each association. But the application has to protect itself for excessive resource usages there too.

7. Acknowledgments

The authors wish to thank Benoit Claise, Julian Cordes, Spencer Dawkins, Gorry Fairhurst, Lennart Grahl, Christer Holmberg, Mirja Kuehlewind, Marcelo Ricardo Leitner, Karen E. Egede Nielsen, Maksim Proshin, Eric Rescorla, Irene Ruengeler, Felix Weinrank, Michael Welzl, Magnus Westerlund, and Lixia Zhang for their invaluable comments.

This work has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 644334 (NEAT). The views expressed are solely those of the author(s).

8. References

8.1. Normative References

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <https://www.rfc-editor.org/info/rfc3758>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <https://www.rfc-editor.org/info/rfc5061>.

[RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, DOI 10.17487/RFC6096, January 2011, <https://www.rfc-editor.org/info/rfc6096>.

[RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, DOI 10.17487/RFC6525, February 2012, <https://www.rfc-editor.org/info/rfc6525>.

[RFC7053] Tuexen, M., Ruengeler, I., and R. Stewart, "SACK-IMMEDIATELY Extension for the Stream Control Transmission Protocol", RFC 7053, DOI 10.17487/RFC7053, November 2013, <https://www.rfc-editor.org/info/rfc7053>.

8.2. Informative References

[I-D.ietf-rtcweb-data-channel] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <https://www.rfc-editor.org/info/rfc6458>.

Authors' Addresses

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
United States

Email: randall@lakerest.net

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Salvatore Loreto
Ericsson
Torshamnsgatan 21
164 80 Stockholm
Sweden

Email: Salvatore.Loreto@ericsson.com

Robin Seggelmann
Metafinanz Informationssysteme GmbH
Leopoldstrasse 146
80804 Muenchen
Germany

Email: rfc@robin-seggelmann.com

Internet Engineering Task Force
INTERNET-DRAFT
Intended Status: Informational
Expires: November 7, 2019

X. Wei
Y. Li
Huawei Technologies
S. Boutros
VMware
L. Geng
China Mobile

May 6, 2019

Tunnel Congestion Feedback
draft-ietf-tsvwg-tunnel-congestion-feedback-07

Abstract

This document describes a method to measure congestion on a tunnel segment based on recommendations from RFC 6040, "Tunneling of Explicit Congestion Notification", and to use IPFIX to communicate the congestion measurements from the tunnel's egress to a controller which can respond by modifying the traffic control policies at the tunnel's ingress.

Table of Contents

1. Introduction

In IP networks, persistent congestion[RFC2914] lowers transport throughput, leading to waste of network resource. Appropriate congestion control mechanisms are therefore critical to prevent the network from falling into the persistent congestion state. Currently, transport protocols such as TCP[RFC793], SCTP[RFC4960], DCCP[RFC4340], have their built-in congestion control mechanisms, and even for certain single transport protocol like TCP there can be a couple of different congestion control mechanisms to choose from. All these congestion control mechanisms are implemented on host side, and there are reasons that only host side congestion control is not sufficient for the whole network to keep away from persistent congestion. For example, (1) some protocol's congestion control scheme may have internal design flaws; (2) improper software implementation of protocol; (3) some transport protocols, e.g. RTP[RFC3550] do not even provide congestion control at all; (4)a heavy load from a much larger than expected number of responsive flows could also lead to persistent congestion.

Tunnels are widely deployed in various networks including public Internet, data center network, and enterprise network etc. A tunnel consists of ingress, egress and a set of intermediate routers. For the tunnel scenario, a tunnel-based mechanism is introduced for network traffic control to keep the network from persistent congestion. Here, tunnel ingress will implement congestion management function to control the traffic entering the tunnel.

This document provides a mechanism of feeding back inner tunnel congestion level to the ingress. Using this mechanism the egress can feed the tunnel congestion level information it collects back to the ingress. After receiving this information the ingress will be able to perform congestion management according to network management policy.

The following subjects are out of scope of current document: it gives no advice on how to select which tunnel endpoints should be used in order to manage traffic over a network criss-crossed by multiple tunnels; if a congested node is part of multiple tunnels, and it causes congestion feedback to multiple traffic management functions at the ingresses of all the tunnels, the draft gives no advice on how all the traffic management functions should respond.

2. Conventions And Terminologies

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]

DP: Decision Point, an logical entity that makes congestion management decision based on the received congestion feedback information.

EP: Enforcement Point, an logical entity that implements congestion management action according to the decision made by Decision Point.

ECT: ECN-Capable Transport code point defined in RFC3168.

3. Congestion Information Feedback Models

The feedback model mainly consists of tunnel egress and tunnel ingress. The tunnel egress composes of meter function and exporter function; tunnel ingress composes EP (Enforcement Point) function, collector function and DP (Decision Point) function.

The Meter function collects network congestion level information, and conveys the information to Exporter which feeds back the information to the collector function.

The feedback message contains CE-marked packet ratio, the traffic volumes of all kinds of ECN marking packets.

The collector collects congestion level information from exporter, after that congestion management Decision Point (DP) function will make congestion management decision based on the information from collector.

The Enforcement Point controls the traffic entering tunnel, and it implements traffic control decision of DP.

```
                        Feedback
              +-----------------------------------+
              |                                   |
              |                                   |
              |                                   V
         +--------------+                  +-------------+
         |  +--------+  |                  | +---------+ |
         |  |Exporter|  |                  | |Collector| |
         |  +---|----+  |                  | +---|-----+ |
         |   +--|--+    |                  |    +|-+     |
         |   |Meter|     |    traffic       |    |DP|    |
         |   +-----+     |<=================|    +--+    |
         |               |                  |    +--+    |
         |               |                  |    |EP|    |
         |               |                  |    +--+    |
         |Egress         |                  |  Ingress   |
         +--------------+                  +-------------+
```

Figure 1: Feedback Model.

4. Congestion Level Measurement

The congestion level measurement is based on ECN (Explicit Congestion Notification) [RFC3168] and packet drop. The network congestion level could be indicated through the ratio of CE-marked packet and the volumes of packet drop, the relationship between these two kinds of indicator is complementary. If the congestion level in tunnel is not high enough, the packets would be marked as CE instead of being dropped, and then it is easy to calculate congestion level according to the ratio of CE-marked packets. If the congestion level is so high that ECT packet will be dropped, then the packet loss ratio could be calculated by comparing total packets entering ingress and total packets arriving at egress over the same span of packets, if packet loss is detected, it could be assumed that severe congestion has occurred in the tunnel.

Egress calculates CE-marked packet ratio by counting different kinds of ECN-marked packet, the CE-marked packet ratio will be used as an indication of tunnel load level. It’s assumed that routers in the tunnel will not drop packets biased towards certain ECN codepoint, so calculating of CE-marked packet ratio is not affect by packet drop.

The calculation of volumes of packet drop is by comparing the traffic volumes between ingress and egress.

Faked ECN-capable transport (ECT) is used at ingress to defer

packet loss to egress. The basic idea of faked ECT is that, when encapsulating packets, ingress first marks tunnel outer header according to RFC6040, and then remarks outer header of Not-ECT packet as ECT, there will be three kinds of combination of outer header ECN field and inner header ECN field: CE|CE, ECT|N-ECT, ECT|ECT (in the form of outer ECN| inner ECN); when decapsulating packets at egress, RFC6040 defined decapsulation behavior is used, and according to RFC6040, the packets marked as CE|N-ECT will be dropped by egress. Faked-ECT is used to shift some drops to the egress in order to calculate CE-marked packet ratio more precisely by egress.

To calculate congestion level, for the same span of packets, the ratio of CE-marked packets will be calculated by egress, and the total bytes count of packets at ingress and egress will be compared to detect the traffic volume loss in tunnel.

The basic procedure of packets loss measurement is as follows:

```
     +-------+                 +------+
     |Ingress|                 |Egress|
     +-------+                 +------+
         |                         |
 +----------------+                |
 |cumulative count|                |
 +----------------+                |
         |                         |
         | <node id-i, ECN counts> |
         |------------------------>|
         |<node id-e, ECN counts>  |
         |<------------------------|
         |                         |
         |                         |
```

Figure 2: Procedure of Packet Loss Measurement

Ingress encapsulates packets and marks outer header according to faked ECT as described above. Ingress cumulatively counts packet bytes for three types of ECN combination (CE|CE, ECT|N-ECT, ECT|ECT) and then the ingress regularly sends cumulative bytes counts message of each type of ECN combination to the egress.

When each message arrives at egress, (1)egress calculates the ratio of CE-marked packet; (2)the egress cumulatively counts packet bytes coming from the ingress and adds its own bytes counts of each type of ECN combination (CE|CE, ECT|N-ECT, CE|N-ECT, CE|ECT, ECT|ECT) to the

message for ingress to calculate packet loss. Egress feeds back CE-marked packet ratio and bytes counts information to the ingress for evaluating congestion level in the tunnel.

The counting of bytes can be at the granularity of the all traffic from the ingress to the egress to learn about the overall congestion status of the path between the ingress and the egress. The counting can also be at the granularity of individual customer's traffic or a specific set of flows to learn about their congestion contribution.

5. Congestion Information Delivery

As described above, the tunnel ingress needs to convey a message containing cumulative bytes counts of packets of each type of ECN combination to tunnel egress, and the tunnel egress also needs to feed back the message of cumulative bytes counts of packets of each type of ECN combination and CE-marked packet ratio to the ingress. This section describes how the messages should be conveyed.

The message travels along the same path with network data traffic, referred as in-band signal. Because the message is transmitted in band, so the message packet may get lost in case of network congestion. To cope with the situation that the message packet gets lost, the bytes counts values are sent as cumulative counters. Then if a message is lost the next message will recover the missing information. Even though the missing information could be recovered, the message should be transmitted in a much higher priority than users' traffic flows.

IPFIX [RFC7011] is selected as a candidate information feedback protocol. IPFIX uses preferably SCTP as transport. SCTP allows partially reliable delivery [RFC3758], which ensures the feedback message will not be blocked in case of packet loss due to network congestion.

Ingress can do congestion management at different granularity which means both the overall aggregated inner tunnel congestion level and congestion level contributed by certain traffic(s) could be measured for different congestion management purpose. For example, if the ingress only wants to limit congestion volume caused by certain traffic(s),e.g UDP-based traffic, then congestion volume for the traffic will be fed back; or if the ingress do overall congestion management, the aggregated congestion volume will be fed back.

When sending message from ingress to egress, the ingress acts as IPFIX exporter and egress acts as IPFIX collector; When feedback congestion level information from egress to ingress, then the egress acts as IPFIX exporter and ingress acts as IPFIX collector.

The combination of congestion level measurement and congestion information delivery procedure should be as following:

The ingress determines IPFIX template record to be used. The template record can be pre-configured or determined at runtime, the content of template record will be determined according to the granularity of congestion management, if the ingress wants to limit congestion volume contributed by specific traffic flow then the elements such as source IP address, destination IP address, flow id and CE-marked packet volume of the flow etc will be included in the template record.

Meter on ingress measures traffic volume according to template record chosen and then the measurement records are sent to egress in band.

Meter on egress measures congestion level information according to template record, the content of template record should be the same as template record of ingress.

Exporter of egress sends measurement record together with the measurement record of ingress back to the ingress.

5.1 IPFIX Extensions

This sub-section defines a list of new IPFIX Information Elements according to RFC7013 [RFC7013].

5.1.1 tunnelEcnCeCeByteTotalCount

Description: The total number of bytes of incoming packets with CE|CE ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD1

Statues: current

Units: bytes

5.1.2 tunnelEcnEct0NectBytetTotalCount

Description: The total number of bytes of incoming packets with ECT(0)|N-ECT ECN marking combination at the Observation Point since

the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD2

Statues: current

Units: bytes

5.1.3 tunnelEcnEct1NectByteTotalCount

Description: The total number of bytes of incoming packets with ECT(1)|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD3

Statues: current

Units: bytes

5.1.4 tunnelEcnCeNectByteTotalCount

Description: The total number of bytes of incoming packets with CE|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD4

Statues: current

Units: bytes

5.1.5 tunnelEcnCeEct0ByteTotalCount

Description: The total number of bytes of incoming packets with CE|ECT(0) ECN marking combination at the Observation Point since the

Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD5

Statues: current

Units: bytes

5.1.6 tunnelEcnCeEct1ByteTotalCount

Description: The total number of bytes of incoming packets with CE|ECT(1) ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD6

Statues: current

Units: bytes

5.1.7 tunnelEcnEct0Ect0ByteTotalCount

Description: The total number of bytes of incoming packets with ECT(0)|ECT(0) ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD7

Statues: current

Units: bytes

5.1.8 tunnelEcnEct1Ect1PacketTotalCount

Description: The total number of bytes of incoming packets with ECT(1)|ECT(1) ECN marking combination at the Observation Point since

the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD8

Statues: current

Units: bytes

5.1.9 tunnelEcnCEMarkedRatio

Description: The ratio of CE-marked Packet at the Observation Point.

Abstract Data Type: float32

ElementId: TBD8

Statues: current

6. Congestion Management

After tunnel ingress receives congestion level information, then congestion management actions could be taken based on the information, e.g. if the congestion level is higher than a predefined threshold, then action could be taken to reduce the congestion level.

The design of network side congestion management SHOULD take host side e2e congestion control mechanism into consideration, which means the congestion management needs to avoid the impacts on e2e congestion control. For instance, congestion management action must be delayed by more than a worst-case global RTT (e.g. 100ms), otherwise tunnel traffic management will not give normal e2e congestion control enough time to do its job, and the system could go unstable.

The detailed description of congestion management is out of scope of this document, as examples, congestion management such as circuit breaker [RFC8084] could be applied. Circuit breaker is an automatic mechanism to estimate congestion, and to terminate flow(s) when persistent congestion is detected to prevent network congestion collapse.

6.1 Example

This subsection provides an example of how the solution described in this document could work.

First of all, IPFIX template records are exchanged between ingress and egress to negotiate the format of data record, the example here is to measure the congestion level for the overall tunnel (caused by all the traffic in tunnel). After the negotiation is finished, ingress sends in-band message to egress, the message contains the number of each kind of ECN-marked packets (i.e. CE|CE, ECT|N-ECT and ECT|ECT) received until the sending of message.

After egress receives the message, the egress calculates CE-marked packet ratio and counts number of different kinds of ECN-marking packets received until receiving the message, then the egress sends a feedback message containing the counts together with the information in ingress's message to ingress.

Figure 3 to Figure 6 below show the example procedure between ingress and egress.

```
+---------------------------------+----------------------+
|Set ID=2                             Length=40          |
|---------------------------------|----------------------|
|Template ID=256                      Field Count =8     |
|---------------------------------|----------------------|
|tunnelEcnCeCeByteTotalCount        Field Length=8       |
|---------------------------------|----------------------|
|tunnelEcnEctNectByteTotalCount     Field Length=8       |
|---------------------------------|----------------------|
|tunnelEcnEctEctByteTotalCount      Field Length=8       |
|---------------------------------|----------------------|
|tunnelEcnCeCeByteTotalCount        Field Length=8       |
|---------------------------------|----------------------|
|tunnelEcnEctNectByteTotalCount     Field Length=8       |
|---------------------------------|----------------------|
|tunnelEcnEctEctByteTotalCount      Field Length=8       |
|---------------------------------|----------------------|
|tunnelEcnCeNectByteTotalCount      Field Length=8       |
|---------------------------------|----------------------|
|tunnelEcnCeEctByteTotalCount         Field Length=8     |
+---------------------------------+----------------------+
|tunnelEcnCEMarkedRatio               Field Length=4     |
+---------------------------------+----------------------+
```

Figure 3: Template Record Sent From Egress to Ingress

```
+---------------------------------+----------------------+
|Set ID=2                              Length=28          |
|---------------------------------|----------------------|
|Template ID=257                       Field Count =3     |
|---------------------------------|----------------------|
|tunnelEcnCeCeByteTotalCount         Field Length=8    |
|---------------------------------|----------------------|
|tunnelEcnEctNectByteTotalCount      Field Length=8    |
|---------------------------------|----------------------|
|tunnelEcnEctEctByteTotalCount       Field Length=8    |
|---------------------------------|----------------------|
```

Figure 4: Template Record Sent From Ingress to Egress

```
 +-------+         +-+  +-+ +-+ +-+  +-+ +-+ +-+  +-------+
 |       |         |M|  |P| |P| |P|  |M| |P| |P|  |       |
 |       |         +-+  +-+ +-+ +-+  +-+ +-+ +-+  |       |
 |       |<-------------------------------------|       |
 |       |                                      |       |
 |       |                                      |       |
 |egress |         +-+             +-+          |ingress|
 |       |         |M|             |M|          |       |
 |       |         +-+             +-+          |       |
 |       |------------------------------------->|       |
 |       |                                      |       |
 |       |                                      |       |
 +-------+                                      +-------+

+-+
|M| : Message Packet
+-+

+-+
|P| : User Packet
+-+
```

Figure 5 Traffic flow Between Ingress and Egress

```
                    Set ID=257, Length=28
       +------+            A1                    +------+
       |      |            B1                    |      |
       |      |            C1                    |      |
       |      |  <----------------------------   |      |
       |      |                                  |      |
       |      |                                  |      |
       |      |      SetID=256, Length=72        |      |
       |      |            A1                    |      |
       |      |            B1                    |      |
       |egress|            C1                    |ingress|
       |      |            A2                    |      |
       |      |            B2                    |      |
       |      |            C2                    |      |
       |      |            D                     |      |
       |      |            E                     |      |
       |      |            R                     |      |
       |      |    ---------------------------->  |      |
       |      |                                  |      |
       +------+                                  +------+
```

Figure 6: Message Between Ingress and Egress

The following provides an example of how tunnel congestion level could be calculated:

Congestion Level could be divided into two categories:(1)slight congestion(no packets dropped); (2)serious congestion (packet dropping happen).

For slight congestion, the congestion level is indicated as the ratio of CE-marked packet:

ce_marked = R;

For serious congestion, the congestion level is indicated as the number of volume loss:

total_ingress = (A1 + B1 + C1)

total_egress = (A2 + B2 + C2 + D + E)

volume_loss = (total_ingress - total_egress)

7. Security Considerations

This document describes the tunnel congestion calculation and feedback.

The tunnel endpoints are assumed to be deployed in the same administrative domain, so the ingress and egress will trust each other, the signaling traffic between ingress and egress will be protected utilizing security mechanism provided IPFIX (see section 11 in RFC7011).

From the consideration of privacy point of view, in case of fine grained congestion management, ingress is aware of the amount of traffic for specific application flows inside the tunnel which seems to be an invasion of privacy. But in any way, the ingress could The solution doesn't introduce more privacy problem.

8. IANA Considerations

This document defines a set of new IPFIX Information Elements (IE),which need to be registered at IANA IPFIX Information Element Registry.

ElementID: TBD1
Name:tunnelEcnCeCePacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with CE|CE ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD2
Name:tunnelEcnEct0NectPacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with ECT(0)|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD3
Name: tunnelEcnEct1NectPacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with

ECT(1)|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD4
Name:tunnelEcnCeNectPacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with CE|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD5
Name:tunnelEcnCeEct0PacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with CE|ECT(0) ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD6
Name:tunnelEcnCeEct1PacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with CE|ECT(1) ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD7
Name:tunnelEcnEct0Ect0PacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with ECT(0)|ECT(0) ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD8
Name:tunnelEcnEct1Ect1PacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter

Status: current
Description:The total number of bytes of incoming packets with ECT(1)|ECT(1)ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD9
Name: tunnelEcnCEMarkedRatio
Data Type: float32
Status: current
Description: The ratio of CE-marked Packet at the Observation Point.

[TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/ipfix/ipfix.xhtml#ipfix-information-elements]

9. References

9.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

[CONEX] Matt Mathis, Bob Briscoe. "Congestion Exposure (ConEx) Concepts, Abstract Mechanism and Requirements", RFC7713, December 2015

9.2 Informative References

[RFC8084] G. Fairhurst. "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-01, April 02, 2015

10. Acknowledgements

Thanks Bob Briscoe for his insightful suggestions on the basic mechanisms of congestion information collection and many other useful comments. Thanks David Black for his useful technical suggestions. Also, thanks Lei Zhu, Lingli Deng, Anthony Chan, Jake Holland, John Kaippallimalil and Vincent Roca for their careful reviews.

Authors' Addresses

Xinpeng Wei
Beiqing Rd. Z-park No.156, Haidian District,
Beijing, 100095, P. R. China
EMail: weixinpeng@huawei.com

Yizhou Li
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China

Phone: +86-25-56624584
EMail: liyizhou@huawei.com

Sami Boutros
VMware, Inc.
EMail: boutross@vmware.com

Liang Geng
China Mobile
EMail: gengliang@chinamobile.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 16, 2016

I. Johansson
Ericsson AB

October 14, 2015

Congestion control for 4G and 5G access
draft-johansson-cc-for-4g-5g-00

Abstract

This memo outlines the challenge that 4G and 5G access brings for transport protocol congestion control and also outlines a few simple examples that can improve transport protocol congestion control performance in 4G and 5G access.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 16, 2016.

Copyright Notice

Table of Contents

1. Introduction

Wireless access is becoming more and more widely used, 4G (LTE) access is one wireless access technology that has built in support for seamless mobility that gives the end user a feeling of being always connected. Transport endpoints may even be unaware of the existence of the 4G access. Everyday use for 4G access includes web, chat, streaming video and lately also WebRTC. These use cases pose different requirements on the underlying access. Evolving existing radio-access technologies like LTE, and new 5G technologies will all be part of a future flexible and dynamic 5G system. 5G has potential to offer lower latency and higher peak throughput. The goal of this document is to provide sufficient input to guide the development of congestion control that is better suited for 4G and 5G access, without an explicit need to know about the presence of 4G or 5G access along the transmission path.

2. The 4G protocol stack impact on transport protocols

This section will go into the different layers in the 4G protocol stack. It will not delve in the very details, recommended reading for more details is found in [LTE]. Rather this section will illustrate what effect each layer can have on the transport protocol efficiency. The description is focused mainly on default radio access bearers, which are commonly used for OTT (Over The Top) services, these bearers typically use Acknowledged Mode (AM), which means that packet loss only occurs as the result of packet drops in AQM (Active Queue manager). Specialized services like VoLTE (Voice over LTE) use different bearer configurations, this is however outside the scope of this document. The concept of bearer is mentioned throughout the document, a bearer is to be seen as a data channel for a given terminal or UE (User Equipment), there could be many bearers with different priorities for a given terminal.

```
------------
|          |
|    IP    |
|          |
------------
|   PDCP   |
------------
|   RLC    |
------------
|   MAC    |
------------
|   PHY    |
------------
```

LTE protocol stack

2.1. PDCP layer

The PDCP (Packet Data Convergence Protocol Layer), ensures that intra-RAT (Radio Access Type) handover, i.e. LTE to LTE is reasonably seamless. The PDCP layer makes sure that packets pending transmission in one cell to terminal connection, are transmitted in the new cell to terminal connection. This way all packets will be ensured to be delivered to the endpoint. PDCP also ensures that all packets are delivered in order up to higher layers.

Packet retransmission typically means that the amount of data to transmit increases immediately after the handover, first the retransmission data needs to be transmitted, then the incoming data needs to be transmitted. Depending on the available link throughput after the handover, an increased RTT may be experienced at a handover

event. In addition, a small temporal delay increase can occur as packet transmission is inhibited at the handover event. Unnecessary, retransmission at handover can be minimized by means of PDCP status reports, but this is not always implemented. Because of the above it is a good practice to keep the amount of data in flight as small as possible, without sacrificing throughput. Excessive amounts of data in flight means potentially more data to retransmit at handover and thus an even more increased RTT.

Packets are typically not lost at handover in a 4G/LTE system. Reliable delivery at handover may however be turned off or is simply not implemented, this means that packets may be lost at handover. The amount of lost packets is then proportional to the number of packets in flight, it is therefore instrumental that the amount of packets in flight is as small as possible, with the objective to reach full link utilization, nothing more. Bufferbloat [REF] can for instance lead to that 1000s of packets are lost at handover, which is of course undesired as it can affect media quality quite considerably.

2.2. RLC layer

The role of the RLC (Radio Link Control) layer is to ensure that packets are delivered in order up to the higher layers, in addition the RLC layer corrects errors that can occur on the MAC layer. The in order delivery constraint means that additional HoL (Head of Line) blocking delay can occur due to errors on the MAC layer.

The throughput on lower layers can vary quite considerably, this manifests itself as a varying size of the available transport. The transport block size depends on how much of the available resources is allocated to a given bearer at a given time instant and also on the channel quality. Because of this the size of the transport blocks can vary from tens of bytes, up to more than 10000 bytes. The rate of change in transport block size also varies with terminal mobility as higher terminal mobility means faster changing channel fading and thus a faster changing channel quality.

For optimal spectrum efficiency, it is important that a sufficient amount of data is available to fill the transport blocks, this data needs to be available already when a bearer is scheduled, in practice within a fraction of a millisecond. To satisfy this requirement, packets need to be queued up and ready for immediate transmission, either on the RLC or the PDCP layer. The transport protocol server (TCP, QUIC) is typically too far away to satisfy this need.

The need to instantly provide sufficient data for optimal spectrum efficiency, given the variability in transport block sizes and

scheduling opportunities for a given bearer, quite naturally leads to a variation in queuing delay and this variation can be larger than what is to be expected from e.g. fixed line access.

The requirement above to have data available can be seen something that contradicts the strive for low latency, and there is indeed a balance to be struck here. What to aim for, maximum throughput or low latency, is something that depends on the requirements from the application. A desire for very low latency comes generally at the cost of reduced peak throughput, this applies to default radio access bearers. QoS classed bearers can have different characteristics and may well be able to deliver both very low latency and high throughput.

2.3. MAC layer

The MAC (Media Access Control) layer handles transmission of transport blocks, the outcome of a transmission attempt can be either success or failure. The signaling of the success is handled with a single ACK/NAK bit. Upon indicated failure, the transport block is retransmitted (with a different channel coding), soft decoding is utilized and the softbits of the first transmission and the retransmission are combined, hopefully with a successful result, if not the case a 3rd retransmission can occur and so on. This is referred to as HARQ (Hybrid Automatic Repeat Request). The maximum number of retransmissions is configurable, if the maximum number of retransmission is reached without successful transmission then the RLC layer will have to handle the retransmission instead.

Errors may also happen in the transmission of the ACK/NAK bit. The event that ACK is decoded as NAK will only lead to that an extra superfluous retransmission occur. The event that a NAK is decoded as an ACK will be handled by the RLC layer as the result of a detected RLC checksum error.

MAC layer retransmissions naturally lead to out of order delivery up to higher layers as some transport blocks are transmitted error free while others need retransmission. The role of the RLC layer is to ensure in order delivery, the effect of this is that HARQ retransmissions and HARQ failures lead to additional delays.

Scheduling of many bearers has the effect that available resources have to be shared between two or more bearers. When new bearers with data to transmit are added in a cell (either handover, or new traffic), it means that the amount of resources need to be shared with an additional party. This can give a large drop in available throughput for already existing users, with the effect of an

increased queuing delay that decreased only when the transmission rate over the bearer is reduced.

The downlink and uplink scheduling differs in some details which are described in the following sub-sections.

2.3.1. Downlink scheduling

Downlink scheduling, or scheduling of packets to terminals, is controlled by the base station. For each TTI (1ms interval) a decision is made on which bearer is to become scheduled, i.e. packets are forwarded from the queue to the terminal in question. The scheduling decision is based on channel quality, and possibly historic bitrate for the given bearers, or it may be just a simple round robin scheduler. The very details of the scheduling algorithms are vendor specific.
DRX (Discontinuous Reception) is a feature implemented to save battery power, in which the terminal sleeps and only checks for the presence on downlink data only at regular intervals.
Given the facts above, downlink data cannot always be transmitted immediately, this has the effect that additional jitter may be added (in the order of 10-20ms). Congestion control algorithms that are tuned with a high sensitivity to delay can by mistake treat this jitter as congestion.

2.3.2. Uplink scheduling

As is the case with downlink scheduling, uplink scheduling is controlled by the base station. A terminal that has data to transmit will first transmit a scheduling request to the base station. The scheduling request does not indicate how many bytes that are in the uplink queue. The base station transmits a scheduling grant back, with a delay that depends on the overall load level. The scheduling grant indicates how many bytes that can be transmitted by the terminal. After this the terminal can transmit the allowed number of bytes, if there is still data in the queue, then a BSR (Buffer Status Report) is attached to the uplink transmission which will in turn trigger an additional scheduling grant from the base station to the terminal and so on until all the data in the queue is transmitted.
The uplink scheduling regime outlined above can break up packet trains, for instance a set of 10 ACKs in the uplink may be broken up to an initial transmission of 2 ACKs followed by the transmission of the remaining 8 ACKs, the HARQ RTT is typically 8ms, which means that the remaining 8 ACKs are delivered upstream 8ms later. This can cause problems for congestion control algorithms that depend on e.g. packet train based estimation of throughput. Also, algorithms that depend on precise RTT estimates may by mistake treat the occurrence above as emerging congestion in the downlink.

This behavior can also trigger coalescing issues similar to those experienced when ACK compression occur.
Worth notice is also that the above effects can occur at low as well as high network load levels.

ACK traffic in uplink can also be delayed due to for instance lack of signaling channel resources for instance if many terminals generate ACK traffic that is so sparse that scheduling requests need to be generated with high frequency. A transport protocol design that tries to limit the amount of ACK traffic can have a performance benefit under such circumstances as the limitation is then more controlled and the protocol can be optimized for this. Reduced ACKs can unfortunately cause coalescing, something that may be mitigated to some extent by means of packet pacing.

3. 4G and 5G evolution

It is currently unclear in what aspect a 5G protocol stack will affect transport protocol performance. Listed below are however some features of evolved 4G and 5G that have relevance in this context:

- Shorter TTIs (Transmission Time Interval) has the potential to reduce the latency. Given that shorter TTIs have impact on the scheduling and also the retransmissions, the impact of shorter TTIs is that errors on the MAC layer will cause less jitter.

- Larger throughput variations can occur as a result of techniques like carrier aggregation and dual connectivity. Carrier aggregation means that additional carriers (possibly in very high frequency bands) are added. Dual connectivity can combine two similar or different radio access technologies on lower layers (below IP). Both technologies mentioned above can lead to large variations in available throughput.

- ECN is specified in 3GPP 36.300 [TS_36300]. ECN can provide with prompt indication of congestion without the need for packet drop caused by normal AQM operation, this can provide with a benefit for e.g. latency sensitive traffic. ECN also gives a explicit indication of congestion, opposed to the implicit congestion indications that loss and delay gives.

The shorter TTI feature is part of the 5G standardization, it should however be stated that the

4. Requirements for improved performance

The above considerations lead to a few things to consider when congestion control is designed for optimal performance in 4G and 5G networks:

- Avoid dependencies on precise RTT estimates: A typical real life case is the Hybrid Slowstart algorithm bundled with the Cubic congestion control. Uplink scheduling can break up transmission of ACKs which will in turn lead to increased RTTs that can falsely be interpreted as congestion.

- Use timestamps: Related to RTT estimate issue above. For instance a modified Hybrid Slowstart algorithm can take timestamp values into account and thus limit the effect of uplink scheduling effects that can distort the transmission of ACKs.

- Minimize latency under load: The quickly changing throughput in 4G/5G calls for a sensible balance between latency and throughput. Some amount of bufferbloat needs to be accepted in order to have enough data to utilize the radio resources optimally and get a high spectrum efficiency, this can however make the reaction to reduced throughput more sluggish. Hybrid loss/delay based congestion control can here be used to find a good balance between latency and throughput.

- ECN support: The transport protocols should support negotiation and use of ECN.

- Faster congestion window increase: Traditional AIMD (Additive Increase Multiplicative Decrease) based congestion avoidance algorithms are too slow to gain the benefits of e.g added carriers, therefore, more high speed alternatives should be considered, that are still reactive to congestion.

- Packet pacing: ACK compression effects can easily occur in 4G/5G networks, packet pacing should be encouraged to mitigate the coalescing effects caused by ACK compression, and will at the same time make ECN detection algorithms more robust.

- ACK reduction: Consider if it is possible to reduce the intensity of acknowledgements, especially in the uplink. Packet pacing may here be beneficial as it can mitigate the coalescing effects that can occur due to reduced ACK intensity.

5. Congestion control examples

This section lists a few examples of algorithm that can be useful

- o Default slowstart algorithms generally only operates at flow start-up and after a retransmission timeout. The drawback with this approach is that the congestion control cannot quickly grab new available capacity due to e.g the addition of an extra carrier. HyStartRestart is a simple add-on to the Hybrid Slowstart algorithm that resumes slowstart if it is detected that the bottleneck is underutilized for a while.

- o Hybrid Cubic borrows the OWD (One Way "extra" Delay) estimation from LEDBAT [RFC6817]. With this addition it is possible to set a target queuing delay that adds a limit to the congestion window based on network queuing delay in addition to the already existing loss based control of the congestion window.

- o Various High Speed congestion control algorithms such as SIAD (Scalable Increase Adaptive Decrease) [TCP_SIAD] can provide with improved performance in the presence of large changes in available throughput resulting from e.g added carriers.

The HyStartRestart and Hybrid Cubic algorithms are described in more detail below. The code samples are shown with the Linux kernel 4.4 version of tcp_cubic.c as basis.

5.1. HyStartRestart

The idea behind HyStartRestart is simply to increase the ssthresh (slow start threshold) if the RTT has been only marginally higher than the min RTT for a number of round trips. The HyStart delay algorithm used for this purpose. Code for this is shown below with the code from Linux tcp_cubic.c as basis

```
Function bictcp_acked(..) is modified according
to the code snippet below
New state variables are added
  u32 last_rtt_high
  u32 last_hyrestart
New constants
  #define N_RTT_LOW 5
  #define N_RTT_HYRESTART 10


/** Old code **/
/* first time call or link delay decreases */
if (ca->delay_min == 0 || ca->delay_min > delay)
   ca->delay_min = delay;


/** New code **/
/*
 * Function bictcp_acked is modified to increase snd_ssthresh when
 * RTT is lower than a given value for a given number of RTTs
 */
if (ca->curr_rtt > ca->delay_min +
    HYSTART_DELAY_THRESH(ca->delay_min >> 3)) {
   ca->last_rtt_high = bictcp_clock();
} else {
   u32 now = bictcp_clock();
   if (now - ca->last_rtt_high > N_RTT_LOW*ca->delay_min &&
       now - ca->last_hyrestart > N_RTT_HYRESTART*ca->delay_min) {
      /* Double ssthresh */
      tp->snd_ssthresh = tp->snd_ssthresh << 1;
   }
}
/** End of new code **/


/** Old code **/
/* hystart triggers when cwnd is larger than some threshold */
if (hystart && tp->snd_cwnd <= tp->snd_ssthresh &&
   tp->snd_cwnd >= hystart_low_window)
      hystart_update(sk, delay);
```

HyStartRestart code

The code above needs to be complemented with a limitation to ssthresh. Furthermore ca->last_hyrestart should be updated to current time whenever a loss or ECN event is detected.

5.2. Hybrid Cubic

The Hybrid Cubic algorithm adds delay sensitivity to the Cubic congestion avoidance algorithm. It is, in the description below assumed that the timestamp option is enabled and that OWD samples are computed, according to the description in LEDBAT [RFC6817]. Furthermore it is assumed than the timestamp clock frequency in sender and receiver are identical or that the sender can infer the timestamp clock frequency of the receiver and recompute timestamp values based on this information.

The function bictcp_update is updated according to the code snippet below.

```
New state variables
  float owd
  u32 last_hycubic_cwnd_reduced
New constants
  #define OWD_TARGET 0.1
  #define OWD_GAIN_UP 100.0
  #define OWD_GAIN_DOWN 0.1



/** New code, inserted before tcp_friendlness: **/
/*
 * The cnt variable is modified depending on the
 * relation between the OWD and the OWD target
 */
if (ca->owd < OWD_TARGET) {
   float tmp = ca->owd/OWD_TARGET;
   int cnt_d = (int) (tmp*OWD_GAIN_UP);
   /*
    * OWD is less than OWD target
    * Increase cnt as OWD is approaching target
    * This will slow down congestion window growth
    * when owd increases
    */
   ca->cnt += cnt_d;
} else {
   /*
    * Set cnt to a low value, this will result in an
    * immediate reduction of CWND
    */
   ca->cnt = 1;
}
/** End of new code **/
```

```
      /* TCP Friendly */
      if (tcp_friendliness) {
         u32 scale = beta_scale;

         delta = (cwnd * scale) >> 3;
         /** New code **/
         if (ca->owd < OWD_TARGET) {
         /** End of new code **/
            while (ca->ack_cnt > delta) {  /* update tcp cwnd */
               ca->ack_cnt -= delta;
               ca->tcp_cwnd++;
            }
         /** New code **/
         } else {
            u32 now = bictcp_clock();
            if (now-ca->last_hycubic_cwnd_reduced > delay) {
               /* At most one reduction per RTT
               float overshoot = (owd-OWD_TARGET)/delay;
               float alpha = MIN(0.5,overshoot*OWD_GAIN_DOWN);
               ca->tcp_cwnd = (int)(ca->tcp_cwnd*(1.0-alpha));
               ca->ssthresh = ca->tcp_cwnd;
               ca->epoch_start = 0;
               ca->last_hycubic_cwnd_reduced = now;
            }

         }
         /** End of new code **/

         if (ca->tcp_cwnd > cwnd) { /* if bic is slower than tcp */
            delta = ca->tcp_cwnd - cwnd;
            max_cnt = cwnd / delta;
            if (ca->cnt > max_cnt)
               ca->cnt = max_cnt;
         }
      }
```

Hybrid Cubic

Note that the code is not fully functional, for instance the floating point arithmetic need to be converted to fixed point ditto.

6. IANA Considerations

This document makes no request of IANA.

7. Security Considerations

The possible outcome of this work has the same possible security considerations as other work around congestion control.

8. Acknowledgements

The following persons have contributed with comments and suggestions for improvements: Kristofer Sandlund, Mats Nordberg, Hans Hannu, Torsten Dudda and Szilveszter Nadas.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

9.2. Informative References

[LTE] "4G: LTE/LTE-Advanced for Mobile Broadband, Second Edition", 2013.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <http://www.rfc-editor.org/info/rfc6817>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

[TCP_SIAD] "TCP SIAD: Congestion Control https://www.ietf.org/proceedings/91/slides/slides-91-iccrg-5.pdf", 2015.

[TS_36300] "3GPP TS 36.300", 2015.

Author’s Address

Ingemar Johansson
Ericsson AB
Laboratoriegraend 11
Luleaa 977 53
Sweden

Phone: +46 730783289
Email: ingemar.s.johansson@ericsson.com

Network Working Group
Internet-Draft
Updates: 3168 (if approved)
Intended status: Experimental
Expires: March 24, 2016

N. Khademi
M. Welzl
University of Oslo
G. Armitage
Swinburne University of Technology
G. Fairhurst
University of Aberdeen
September 21, 2015

TCP Alternative Backoff with ECN (ABE)
draft-khademi-alternativebackoff-ecn-01

Abstract

This memo provides an experimental update to RFC3168. It updates the TCP sender-side reaction to a congestion notification received via Explicit Congestion Notification (ECN). ECN-marking can allow a network device to signal congestion at a point before a transport experiences congestion loss or additional queueing delay. The updated method is less conservative than the TCP reaction in response to loss. The intention is to achieve good throughput when the queue at the bottleneck is smaller than the bandwidth-delay-product of the connection. This is more likely when an Active Queue Management (AQM) mechanism has used ECN to CE-mark a packet, than when a packet was lost. Future versions of this document will discuss SCTP as well as other transports using ECN.

Status of This Memo

Table of Contents

1. Introduction

Explicit Congestion Notification (ECN) is specified in [RFC3168]. It allows a network device that uses Active Queue Management (AQM) to set the congestion experienced, CE, codepoint in the ECN field of the IP packet header, rather than to drop ECN-capable packets when incipient congestion is detected. When an ECN-capable transport is used over a path that supports ECN, it provides the opportunity for flows to improve their performance in the presence of incipient congestion [I-D.AQM-ECN-benefits].

[RFC3168] not only specifies the router use of the ECN field, it also specifies a TCP procedure for using ECN. This states that a TCP sender should treat the ECN indication of congestion in the same way as that of a non-ECN-Capable TCP flow experiencing loss, by halving

the congestion window "cwnd" and by reducing the slow start threshold "ssthresh". [RFC5681] stipulates that TCP congestion control sets "ssthresh" to max(FlightSize / 2, 2*SMSS) in response to packet loss. Consequently, a non-ECN enabled standard TCP flow using this reaction needs significant network queue space: it can only fully utilize a bottleneck when the length of the link queue (or the AQM dropping threshold) is at least the bandwidth-delay product (BDP) of the flow.

A backoff multipler of 0.5 (halving cwnd and sshthresh after packet loss) is not the only available strategy. As defined in [ID.CUBIC], CUBIC multiplies the current cwnd by 0.8 in response to loss (although the Linux implementation of CUBIC has used a multiplier of 0.7 since kernel version 2.6.25 released in 2008). Consequently, CUBIC flows more fully utilize paths even when the bottleneck queue is slightly shorter than the bandwidth-delay product of the flow. However, in the case of a DropTail (FIFO) queue without AQM, such less-aggressive backoff increases the risk of creating a standing queue [CODEL2012].

Devices implementing AQM are likely to be the dominant (and possibly only) source of ECN CE-marking for packets from ECN-capable senders. AQM mechanisms typically strive to maintain a small queue length, regardless of the bandwidth-delay product of flows passing through them. Receipt of an ECN CE-mark might therefore reasonably be taken to indicate that a small bottleneck queue exists in the path, and hence the TCP flow would benefit from using a less aggressive backoff multiplier.

Results reported in [ABE2015] show significant benefits (improved throughput, resulting in reduced completion times for short flows) when reacting to ECN-Echo by multiplying cwnd and sstthresh with a value in the range [0.7..0.85]. Section 2 describes the rationale for this change. Section 3 specifies a change to the TCP sender backoff behaviour in response to an indication that CE-marks have been received by the receiver.

2. Discussion

Much of the background to this proposal can be found in [ABE2015]. Using a mix of experiments, theory and simulations with standard NewReno and CUBIC, [ABE2015] recommends enabling ECN and "...letting individual TCP senders use a larger multiplicative decrease factor in reaction to ECN CE-marks from AQM-enabled bottlenecks." Such a change is noted to result in "...significant performance gains in lightly-multiplexed scenarios, without losing the delay-reduction benefits of deploying CoDel or PIE."

2.1. Why use ECN to vary the degree of backoff?

The classic rule-of-thumb dictates a BDP of bottleneck buffering if a TCP connection wishes to optimise path utilisation. A single TCP connection running through such a bottleneck will have opened cwnd up to 2*BDP by the time packet loss occurs. [RFC5681]’s halving of cwnd and ssthresh pushes the TCP connection back to allowing only a BDP of packets in flight -- just enough to maintain 100% utilisation of the network path.

AQM schemes like CoDel and PIE use congestion notifications to constrain the queuing delays experienced by packets, rather than in response to impending or actual bottleneck buffer exhaustion. With current default delay targets, CoDel and PIE both effectively emulate a shallow buffered bottleneck (section II, [ABE2015]). This interacts acceptably for TCP connections over low BDP paths, or highly multiplexed scenarios (lmany concurrent TCP connections). However, it interacts badly with lightly-multiplexed cases (few concurrent connections) over high BDP paths. Conventional TCP backoff in such cases leads to gaps in packet transmission and underutilisation of the path.

In an ideal world, the TCP sender would adapt its backoff strategy to match the effective depth at which a bottleneck begins indicating congestion. In the practical world, [ABE2015] proposes using the existence of ECN CE-marks to infer whether a path’s bottleneck is AQM-enabled (shallow queue) or classic DropTail (deep queue), and adjust backoff accordingly. This results in a change to the requirements of [RFC3168], which required TCP senders to respond the same following indication of a received ECN CE-mark and a packet loss, making these equivalent signals of congestion. (The idea to change this behaviour pre-dates ABE. [ICC2002] also proposed using ECN CE-marks to modify TCP congestion control behaviour, using a larger multiplicative decrease factor in conjunction with a smaller additive increase factor to deal with RED-based bottlenecks that were not necessarily configured to emulate a shallow queue.)

[RFC7567] states that "deployed AQM algorithms SHOULD support Explicit Congestion Notification (ECN) as well as loss to signal congestion to endpoints" and [I-D.AQM-ECN-benefits] encourages this deployment. Apple recently announced their intention to enable ECN in iOS 9 and OS X 10.11 devices [WWDC2015]. By 2014, server-side ECN negotiation was observed to be provided by the majority of the top million web servers [PAM2015], and only 0.5% of websites incurred additional connection setup latency using RFC3168-compliant ECN-fallback mechanisms.

2.2. Choice of ABE multiplier

ABE decouples a TCP sender's reaction to loss and ECN CE-marks. The description respectively uses beta_{loss} and beta_{ecn} to refer to the multiplicative decrease factors applied in response to packet loss and in response to an indication of a received CN CE-mark on an ECN-enabled TCP connection (based on the terms used in [ABE2015]). For non-ECN-enabled TCP connections, no ECN CE-marks are received and only beta_{loss} applies.

In other words, in response to detected loss:

```
cwnd_(n+1) = cwnd_n * beta_{loss}
```

and in response to an indication of a received ECN CE-mark:

```
cwnd_(n+1) = cwnd_n * beta_{ecn}
```

The higher the values of beta_*, the less aggressive the response of any individual backoff event.

The appropriate choice for beta_{loss} and beta_{ecn} values is a balancing act between path utilisation and draining the bottleneck queue. More aggressive backoff (smaller beta_*) risks underutilising the path, while less aggressive backoff (larger beta_*) can result in slower draining of the bottleneck queue.

The Internet is already running with at least two different beta_{loss} values, [RFC5681]'s 0.5, and Linux CUBIC's 0.7. ABE proposes no change to beta_{loss} used by any current TCP implementations.

beta_{ecn} depends on how we want to optimise the reponse of a TCP connection to shallow AQM marking thresholds. beta_{loss} reflects the preferred response of each TCP algorithm when faced with exhaustion of buffers (of unknown depth) signalled by packet loss. Consequently, for any given TCP algorithm the choice of beta_{ecn} is likely to be algorithm-specific, rather than a constant multiple of the algorithm's existing beta_{loss}.

A range of experiments (section IV, [ABE2015]) with NewReno and CUBIC over CoDel and PIE in lightly multiplexed scenarios have explored this choice of parameter. These experiments indicate that CUBIC connections benefit from beta_{ecn} of 0.85 (cf. beta_{loss} = 0.7), and NewReno connections see improvements with beta_{ecn} in the range 0.7 to 0.85 (c.f., beta_{loss} = 0.5).

3. Updating the Sender-side ECN Reaction

This section specifies an experimental update to [RFC3168].

3.1. RFC 2119

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3.2. Update to RFC 3168

This document specifies an update to the TCP sender reaction that follows when the TCP receiver signals that ECN CE-marked packets have been received.

The first paragraph of Section 6.1.2, "The TCP Sender", in [RFC3168] contains the following text:

"If the sender receives an ECN-Echo (ECE) ACK packet (that is, an ACK packet with the ECN-Echo flag set in the TCP header), then the sender knows that congestion was encountered in the network on the path from the sender to the receiver. The indication of congestion should be treated just as a congestion loss in non-ECN-Capable TCP. That is, the TCP source halves the congestion window "cwnd" and reduces the slow start threshold "ssthresh"."

This memo updates this by replacing this with the following text:

"If the sender receives an ECN-Echo (ECE) ACK packet (that is, an ACK packet with the ECN-Echo flag set in the TCP header), then the sender knows that congestion was encountered in the network on the path from the sender to the receiver. The indication of congestion SHOULD induce a less conservative reaction than loss: the TCP source multiplies the congestion window ’cwnd’ with 0.8 and reduces the slow start threshold ’ssthresh’."

3.3. Status of the Update

XXX Author’s note: Once ICCRG evalaution has been completed an appropriate outcome may be inserted here XXX

The congestion control behaviour specified in this update will be evaluated by the IRTF Internet Congestion Control Research Group (ICCRG), to determine whether it is thought safe for deployment in the general Internet.

XXX Author’s note: If this is adopted for publication as an Experimental RFC we need to explain why this is not PS XXX

The present specification has been assigned an Experimental status, because this is common practice for first introduction of changes to the TCP protocol specification, where deployment experience is usually required prior to publishing a Standards-Track document.

This update is a sender-side only change. Like other changes to congestion-control algorithms it does not require any change to the TCP receiver or to network devices (except to enable an ECN-marking algorithm [RFC3168] [RFC7567]). If the method is only deployed by some TCP senders, and not by others, the senders that use this method can gain advantage, possibly at the expense of other flows that do not use this updated method. This advantage applies only to ECN-marked packets and not to loss indications. Hence, the new method can not lead to congestion collapse.

4. Acknowledgements

Authors N. Khademi, M. Welzl and G. Fairhurst were part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed are solely those of the authors.

The authors would like to thank the following people for their contributions to [ABE2015]: Chamil Kulatunga, David Ros, Stein Gjessing, Sebastian Zander.

5. IANA Considerations

XX RFC ED - PLEASE REMOVE THIS SECTION XXX

This memo includes no request to IANA.

6. Security Considerations

The described method is a sender-side only transport change, and does not change the protocol messages exchanged. The security considerations of RFC 3819 therefore still apply.

This document describes a change to TCP congestion control that can make TCP senders more aggressive than flows using TCP as specified in RFC 3819. This could lead to a change in the capacity achieved by flows sharing a network bottleneck. If some flows use this method and share capacity with other flows using previous methods this could reduce fairness in the capacity allocation. Similar unfairness is also exhibited by other congestion control mechanisms that have been

in use in the Internet for many years (e.g., CUBIC [ID.CUBIC]). Unfairness may also be a result of other factors, including the round trip time experienced by a flow. This advantage applies only to ECN-marked packets and not to loss indications, and will therefore can not lead to congestion collapse.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

7.2. Informative References

[ABE2015] Khademi, N., Welzl, M., Armitage, G., Kulatunga, C., Ros, D., Fairhurst, G., Gjessing, S., and S. Zander, "Alternative Backoff: Achieving Low Latency and High Throughput with ECN and AQM", CAIA Technical Report CAIA-TR-150710A, Swinburne University of Technology, July 2015, <http://caia.swin.edu.au/reports/150710A/CAIA-TR-150710A.pdf>.

[CODEL2012] Nichols, K. and V. Jacobson, "Controlling Queue Delay", July 2012, <http://queue.acm.org/detail.cfm?id=2209336>.

[I-D.AQM-ECN-benefits] Fairhurst, G. and M. Welzl, "The Benefits of using Explicit Congestion Notification (ECN)", Internet-draft, IETF work-in-progress draft-ietf-aqm-ecn-benefits-05, June 2015.

[ICC2002] Kwon, M. and S. Fahmy, "TCP Increase/Decrease Behavior with Explicit Congestion Notification (ECN)", IEEE ICC 2002, New York, New York, USA, May 2002, <http://dx.doi.org/10.1109/ICC.2002.997262>.

[ID.CUBIC] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", Internet-draft, IETF work-in-progress draft-ietf-tcpm-cubic-00, June 2015.

[PAM2015] Trammell, B., Kuhlewind, M., Boppart, D., Learmonth, I., Fairhurst, G., and R. Scheffenegger, "Enabling Internet-wide Deployment of Explicit Congestion Notification", Proceedings of the 2015 Passive and Active Measurement Conference, New York, March 2015, <http://ecn.ethz.ch/ecn-pam15.pdf>.

[WWDC2015] Lakhera, P. and S. Cheshire, "Your App and Next Generation Networks", Apple Worldwide Developers Conference 2015, San Francisco, USA, June 2015, <https://developer.apple.com/videos/wwdc/2015/?id=719>.

Authors' Addresses

Naeem Khademi
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Email: naeemk@ifi.uio.no

Michael Welzl
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Email: michawe@ifi.uio.no

Grenville Armitage
Centre for Advanced Internet Architectures
Swinburne University of Technology
PO Box 218
John Street, Hawthorn
Victoria 3122
Australia

Email: garmitage@swin.edu.au

Godred Fairhurst
University of Aberdeen
School of Engineering, Fraser Noble Building
Aberdeen AB24 3UE
UK

Email: gorry@erg.abdn.ac.uk

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 21, 2016

K. Nielsen
R. De Santis
Ericsson
A. Brunstrom
Karlstad University
M. Tuexen
Muenster Univ. of Appl. Science
R. Stewart
Netflix, Inc.

October 19, 2015

SCTP Tail Loss Recovery Enhancements
draft-nielsen-tsvwg-sctp-tlr-02.txt

Abstract

Loss Recovery by means of T3-Retransmission has significant detrimental impact on the delays experienced through an SCTP association. The throughput achievable over an SCTP association also is negatively impacted by the occurrence of T3-Retransmissions. The present SCTP Fast Recovery algorithms as specified by [RFC4960] are not able to adequately or timely recover losses in certain situations, thus resorting to loss recovery by lengthy T3-Retransimissions or by non-timely activation of Fast Recovery. In this document we specify a number of enhancements to the SCTP Loss Recovery algorithms which amends some of these deficiencies with a particular focus on Loss Recovery for drops in Traffic Tails. The enhancements supplement the existing algorithms of [RFC4960] with proactive probing and timer driven activation of the Fast Retransmission algorithm as well as a number of enhancements of the Fast Retransmission algorithm in itself are specified.

Status of This Memo

This Internet-Draft will expire on April 21, 2016.

Copyright Notice

Table of Contents

1. Introduction

Loss Recovery by means of T3-Retransmission has significant impact on the delays experienced through, as well as, the throughput achievable over an SCTP association. Loss Recovery by Fast Retransmission operation in many situations is superior to T3-Retransmission from both a latency and a throughput perspective.

The present SCTP Fast Retransmission algorithm, as specified by [RFC4960], is driven uniquely by exceed of a DupTresh number of miss indication counts stemming for returned SACKs, and it is as such not able to adequately or timely recover losses in traffic tails where a sufficient number of such SACKs may not be generated, there resorting to loss recovery by T3-Retransimissions or by non-timely activation of Fast Recovery. Non-timely activation here refer to the situation where activation of Fast Recovery for packets lost within one data burst needs to await arrival of SACKs from a subsequent data burst.

By drop in traffic tails (or tail drops) we refer generally and specifically to the following situations:

1. Drops of the last SCTP packets of an SCTP association or more generally drop of packets in the end of an SCTP association which are not proceeded by more than DupThresh number of packets which are not dropped.

2. Drops among packets sent in a the end of bursts spaced by pauses of time equal to or greater than the T3-timeout (approximately). It is noted that such bursts (pauses in between bursts) may result from application limitations, from congestion control limitations or from receiver side limitations.

3. Drops among packets sent so sparsely that each dropped packet constitutes a tail drop in that DupThresh number of packets would not be sent (would not be available for sent) prior to expiry of the T3-timeout.

It shall be noted that while the above traffic drop criteria describe drops among the forward data packets only, then drops among forward data packets combined with drops of the returned SACKs may together result in that an insufficient number of SACKs be returned to traffic sender for that the Fast Retransmission algorithm be activated prior to T3-timeout occurring. The tail traffic situations for which SCTP Fast Retransmission is not able to recover the losses is thus in general broader than the exact situations listed above. The improvements specified include enhancement of SCTP to deduce the miss indication counts from enhanced scoreboard information thus removing some of the vulnerability of the present SCTP miss indication counting to loss of SACKs.

1.1. The SCTP TLR Function

The function proposed for enhancements of the SCTP Loss Recovery operation for Traffic Tail Losses is divided in two parts:

- o Enhancements of SCTP Fast Retransmission (SCTP FR) algorithm by means of the following Tail Loss Recovery improving functions inspired by or specified by [RFC6675] for TCP:

  - * miss indication counting for a missing (non-SACK’ed) TSN will be based on augmented scoreboard information such that the miss indications will be based not on the number of returned SACKs but on the number of SACK’ed SCTP packets carrying data chunks of higher TSNs. The mechanism is specified both in terms of packets, the book-keeping of which requires new logic, as well as in terms of a less implementation demanding byte based variant following the Islost() approach of [RFC6675]. We shall refer to this improvement as Extended miss indication Counting.

  - * Fast Recovery operation is extended to include the "last resort" retransmission, Nextseg 3) and Nextseg 4), operations of [RFC6675], thus supporting conditional proactive fast retransmissions of missing, but not yet classified as lost, TSNs within the Fast Recovery Exit Point.

- o New SCTP Tail Loss Recovery State machine with proactive timer driven activation of (the enhanced) Fast Recovery operation. Timer driven activation of Fast Recovery is initiated for outstanding data whenever a certain time, shorter then the T3 timeout, has elapsed from the transmittal of the lowest outstanding TSN and network responsiveness, in form of SACKs of packets ahead of the TSN, has been proven since the transmittal of the lowest outstanding TSN. The SCTP TLR mechanism implements a new timer, the Tail Loss Probe timer (PTO), and it works in parts by:

* Forced activation of Fast Recovery when network responsiveness has been proven, and the PTO timer has kicked, since transmittal of the lowest outstanding TSN, but additional traffic sent (SACKs of TSNs ahead of the TSN) has not served to activate Fast Recovery based on the Extended Mis Indication Counting.

* Probing for network responsiveness, by transmittal of a TLR probe packet, when no network responsiveness information (no SACKs have been received for any packets ahead of line of the TSN) is available at expiration of the PTO timer relative to the lowest outstanding TSN

* Activation for T3-retransmission Loss Recovery only when the network remains unresponsive (no SACKs are received) also after transmittal, and subsequently timeout, of a TLR probe packet.

1.1.1. Dependencies

The SCTP TLR procedures proposed apply as add-on supplements to any SCTP implementation based on [RFC4960]. The SCTP TLR procedures in their core are sender-side only and do not impact the SCTP receiver.

Exploitation of SCTP immediate SACK feature, [RFC7053], and usage of new (to be defined) Unambiguous Selective Acknowledgement feature of SCTP require support in both sender and receiver of these SCTP extensions.

1.2. Relation to other work

1.2.1. Early Retransmit and RTO Restart

It is noted that the Early Retransmit algorithm, [RFC5827], addresses activation of Fast Recovery for a particular subset of the tail drop situations in target of the SCTP TLR function. The solution proposed embeds (as a special case) the Early Retransmits algorithm in the delayed variant, experienced with for TCP in [DUKKIPATI02] in which Early Retransmission is only activated provided a certain time has elapsed since the lowest outstanding TSN was transmitted. The delay adds robustness towards spurious retransmissions caused by "mild" packet re-ordering as documented for TCP in [DUKKIPATI02].

It is further noted that depending on the exact situation (e.g., drop pattern, congestion window and amount of data in flight) then T3-retransmission procedures need not be inferior to Fast Retransmission procedures. Rather in some situations T3-retransmission will indeed be superior as T3-retransmissions allow for ramp up of the congestion window during the recovery process.

The changes proposed in this document focus on improving the Loss Recovery operation of SCTP by enforcing timely activation of (improved) Fast Retransmission algorithms. With the purpose to reduce the latency of the TCP and SCTP Loss Recovery operation [HURTIG] has taken the alternative approach of accelerating the activation of T3-retransmission processes when Fast Recovery is not able to kick in to recover the loss. [HURTIG] only addresses a subset of the Tail loss scenarios in scope in the work presented here. The ideas of [HURTIG] for accurate RTO restart are drawn on in the solution proposed here for accurate restart of the new tail loss probe timer (PTO-timer) as well as for accurate set of the T3-timer under certain conditions thus harvesting some of the same latency optimizations as [HURTIG]. The same approach has recently been exploited for TCP by the invention of the TLPR function by the authors of [Rajiullah].

1.2.2. TCP applicability

SCTP Loss Recovery operation in its core is based on the design of Loss Recovery for TCP with SACK enabled. The enhancements of SCTP Tail Loss Recovery proposed here are applicable for TCP.

Note: The - to be determined - exploitation of SCTP immediate SACK feature, [RFC7053], and the - to be determined - usage of new unambiguous selective acknowledgement feature of SCTP may not be readably applicable to TCP at present. ISSUE: Need to follow up on [zimmermann02], [zimmermann03],

It is noted that while the SCTP TLR algorithms and SCTP TLR state machine defined is inspired by the timer driven tail loss probe approach specified in [DUKKIPATI01] for TCP, then the solution defined here differs in the approach taken. The approach here is a clean state approach defining a new comprehensive SCTP TLR state machine as an add-on to the (at least conceptually) existing Fast Recovery and T3-Retransmission SCTP state machines of SCTP. Thereby the SCTP TLR algorithm is able to address all tail loss patterns, whereas the approach of [DUKKIPATI01] relies on a number of experimental mechanisms ([DUKKIPATI02], [MATHIS], [RFC5827]) defined for TCP in IETF or in Research with ad hoc extension to support selected tail loss patterns by addition of the tail loss probe mechanism and the therefrom driven activation of the mechanisms.

1.2.3. Packet Re-ordering

The solution proposed is an enhancement of the existing mis indication counting based Fast Recovery operation of SCTP, [RFC4960], and as such the solution inherits the fundamental vulnerability to

packet re-ordering that the SCTP Fast Retransmission algorithm of [RFC4960] embeds.

For deployment of SCTP in environments where the Fast Retransmission algorithm of [RFC4960] gives rise to spurious entering of Fast Recovery it would be relevant to look into remedies which may detect such and undo the effects of such. Possibly following the approaches taken for TCP (and SCTP) in this area.

OPEN ISSUE: In severe packet re-ordering situations where the second packet of two subsequently sent packets outrace the first packet in arrival with more than PTO time, then such may tricker the SCTP TLR function to enter spurious Fast Recovery. It is conjectured that the this situation does not significantly increase the vulnerability of Loss Recovery to packet-reordering. To be determined and evaluated.

1.2.4. Congestion Control

In its very nature of prompting for activation of Fast Recovery instead of T3-Retransmission Recovery then the benefit of the solution proposed versus the existing solution of [RFC4960] will depend on the CC operation not only during the recovery process but also after exit of the recovery process. In this context it is noted that the prior approach taken for TCP, [DUKKIPATI01], has been documented for a TCP implementation running CUBIC, e.g., see [zimmermann01], whereas SCTP runs a CC algorithm more similar to TCP Reno CC as defined by [RFC5681].

The solution at present is defined within the constraints of existing Congestion Control principles of STCP as defined by [RFC4960]. It is anticipated that Congestion Control improvements are desirable for SCTP in general as well as for the functions defined here in particular.

1.2.5. CMT-SCTP Applicability

The SCTP TLR specification in this document applies to a SCTP implementation following the [RFC4960] principles of using one shared SACK clock spanning the data transfer over multiple paths. It is noted that in its nature of maintaining the common SACK clock principles of [RFC4960] then the SCTP TLR mechanism specified here retains some of the vulnerabilities from [RFC4960] to spurious (or delayed) entering of Fast Recovery operation caused by path changes in inhomogeneous environments (change of data transfer among paths of significantly different RTTs). The validity of this choice is motivated by that concurrent data transfer on multiple paths is the exception case in [RFC4960] MH SCTP and remains the exception also with the enhancements of [RFC4960] specified here.

It is envisaged that the SCTP TLR mechanism specified is readably applicable also to a SCTP implementation supporting concurrent multi path transfer in line with the specification of [CMT-SCTP]. Though is it emphasized that SCTP-TLR, when applied to [CMT-SCTP], needs some adjustments as it should be applied in a split manner following the principles of SFR of [CMT-SCTP].

2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

For the purposes of defining the SCTP TLR function, we use the following terms and concepts:

"DupThresh": The number of miss indication counts on an outstanding TSN at the reach of which SCTP declares the TSN as lost and enters Fast Recovery for the TSN if not in Fast Recovery already.

"Flight size": At any given time we define the "Flight size" to be the number of bytes that a SCTP sender considers to be in flight in the network from the sender to the receiver. It is noted that the bytes of a message, which is considered lost and which has not been retransmitted, is not contained in the Flight size. Further it is noted that the bytes of a message which has been retransmitted (once) will count either once or twice in the Flight size depending on whether SCTP considers the first transmission of the message as having been lost (dropped) in the network.

"Outstanding TSN": A TSN (and the associated DATA chunk) that has been sent by the SCTP sender for which it has not yet received an acknowledgement and which the SCTP sender has not abandoned (e.g., abandoned as a result of [RFC3758]).

"highTSN": The highest outstanding TSN at this point in time.

"lowTSN": The lowest outstanding TSN at this point in time.

"Scoreboard": An SCTP sender need maintain a data structure to store various information on a per outstanding TSN basis. This includes the selective acknowledgment information, miss indication counts, bytes counts and other information defined [RFC4960], in this document and in other SCTP specifications. This data structure we refer to as "scoreboard". The specifics of the scoreboard data structure are out of scope for this document (as

long as the implementation can perform all functions required by this specification).

3. Description of Algorithms

3.1. SCTP Scoreboard and miss indication Counting Enhancement

Entering of Fast Recovery in SCTP, as specified by [RFC4960]), is driven by miss indication counts. When a TSN has received DupThresh=3 miss indication counts, the TSN is declared lost and will be eligible for fast retransmission via Fast Recovery procedure.

miss indication counts are in RFC4960 SCTP driven entirely by receipt of SACKs in accordance with the Highest TSN Newly Acknowledged algorithm (section 7.2.4 of [RFC4960]):

> Highest TSN Newly Acknowledged (HTNA): For each incoming SACK, miss indications are incremented only for missing TSNs prior to the highest TSN newly acknowledged in the SACK. A newly acknowledged DATA chunk is one not previously acknowledged in a SACK.

An evident issue with the HTNA algorithm is that it is vulnerable to loss of SACKs. In many situations loss of SACKs will result only in a slight delayed entering of Fast Recovery for a dropped TSN, but generally, then by relying on HTNA algorithm only, loss of SACKs will further broaden the traffic tails situations where Fast Recovery either not be activated in a timely manner or not be activated at all due to the receipt of an insufficient number SACKs only.

In order to make SCTP Fast Recovery more robust towards drop of SACKs, the following extension of the HTNA algorithm SHOULD be supported by an SCTP implementation:

> Newly Acked Packets ahead-of-line (NAPahol): For each incoming SACK, miss indications are incremented only for missing TSNs prior to the highest TSN newly acknowledged in the SACK. A newly acknowledged DATA chunk is one not previously acknowledged in a SACK. For each missing TSN thus potentially eligible for additional miss indication counts, the number of miss indications to be given shall follow the number of newly acknowledged packets ahead of line of the packet of the missing TSN.

The solution is robust towards split SACK. The solution requires for the SCTP implementation to keep track of the relationship in between data chunks (TSN numbers) and packets. One solution is for the SCTP implementation to maintain a packet id as a monotonically incrementing packet sequence number to map chunks to packets and for

each outstanding chunk to keep state of the packet id that the chunk was sent in as well as (incrementally updated) the packet ids of up to DupThresh-1 (=2) packets ahead of line for which chunks have been SACKed.

For accurate PTO-timer management, using the restart principles of [HURTIG] and [Rajiullah], see Section 3.3, an SCTP TLR implementation is required to keep track of the time at which packets/TSNs are transmitted (or strictly speaking to be able to deduce the time since a packet/a TSN was last transmitted). An implementation may exploit timestamps for the generation of (part of) the packet id as well as for the mentioned time management thereby limiting the additional overhead required for the packet id storage.

As an alternative to the above accurate packet counting then an SCTP implementation MAY, to reduce implementation complexity, instead support the following bytes counting based extension of the RFC4960 HTNA algorithm:

> Highest Bytes Newly Acknowledged (HBNA): For each incoming SACK, miss indications are incremented only for missing TSNs prior to the highest TSN newly acknowledged in the SACK. A newly acknowledged DATA chunk is one not previously acknowledged in a SACK. For each missing TSN thus eligible for additional mis indication counts, the number of miss indications to be given shall follow the number of newly acknowledged bytes in the SACK ahead of line of the missing TSN in the following manner Add-miss indication-count(TSN) = Ceiling((Newly bytes ahead of line(TSN))/PMTU).

The HBNA approach as specified above is vulnerable to split of SACK. An implementation choice which is robust to split of SACK is to recalculate the total amount of selectively acknowledged bytes ahead of line of an outstanding TSN and update the miss indication count of the TSN as Ceiling((Selectively Acked bytes ahead of line (TSN))/PMTU). This more robust implementation choice however demands either for maintain of additional state per TSN, namely the Selectively Acked bytes ahead of line (TSN) or for extensive repeated computations. Risk of split SACK may not be weighty enough to worth such implementation complexity.

The HBNA approach follows the approach taken for TCP, Islost(), in [RFC6675]. It is noted, however, that due to the message based approach of SCTP, then a byte based approach generally will be less accurate as a measure for the number of packet received ahead of line than it is for byte stream based TCP.

3.1.1. Multi-Path Considerations

In multi-homed [RFC4960] SCTP, data that potentially will be subject to fast retransmission may be in flight on multiple paths. This (exception) situation can occur as a result of a change of the data transfer path, which may come about, e.g., as a result of a switchback operation performed autonomously by SCTP or as a result of a management operation setting a new primary path. The situation can also occur as a result of destination directed data transfer where the destination address specified is different from the present data transfer path destination. In an [RFC4960] SCTP implementation, SACKs of data sent on one path will increase the miss indication counts of data with lower TSN in flight on a different path. As such SACKs of data sent on one path may actually result in generation of (potentially spurious) loss event reactions on a different path. This fundamental aspect of [RFC4960] miss indication counting is not changed in this document. Meaning that it is not intended for the miss indication counting improvements defined above, i.e., the NAPahol and the HBNA mechanisms, to discriminate among the paths on which the SACK’ed data contributing to the miss indication counting has been sent.

3.2. RFC6675 nextseg() Tail Loss Enhancements for SCTP FR

The Fast Retransmission algorithm for TCP as specified in [RFC6675] implements some differences compared to the Fast Retransmission algorithm specified for SCTP by [RFC4960]. Of particular significance for recovery of losses in traffic tail scenarios are the fact that the [RFC6675] algorithm, once Fast Recovery has been activated, takes two "last resort" retransmission measures, step 3) and step 4) of Nextseg() of [RFC6675]. These measures facilitate the recovery of losses in situations where only an insufficient number of SACKs would be able to be generated to complete the Fast Recovery process without resorting to T3-timeout. For SCTP Fast Recovery we formulate the equivalent measures as follows:

Last Resort Retransmission: If the following conditions are met:

* there are no outstanding TSN’s eligible for fast retransmission due to DupThresh or more miss indications

* there is no new data available for transmission

then an outstanding TSN less than or equal to the Fast Recovery Exit Point, for which there exists SACKs of chunks ahead of line of the TSN, may be retransmitted provided the CWND allow. The bytes of a TSN which is retransmitted in this manner are not subtracted from the Flight size prior to this action be taken nor

as a result of this action. If the miss indication count of the TSN subsequently reaches the DupThresh value, the bytes of the TSN shall be subtracted from the Flight size. Once acknowledged the remaining contribution of this TSN in the Flight size (whether it be there counted once or twice at this point in time) is subtracted. A TSN which is retransmitted in this manner will be marked as ineligible for a subsequent fast retransmit (see considerations on Multiple Fast Retransmission operation in Section 3.3.1.3).

An SCTP implementation which implements the Unambiguous SACK feature of Appendix A may implement a more accurate calculation of the flightsize when doing Last Resort Retransmission. That is, instead of subtracting the contribution from the retransmitted TSN from the flightsize once the acknowledgement of the TSN arrives, the SCTP implement may distinguish where the acknowledgment is for the original TSN or for the retransmitted TSN and in case the acknowledgement is not for the retransmitted TSN, SCTP should delay the subtract of the bytes of the retransmitted TSN from the flightsize until either an acknowledgement of the retransmitted TSN is received (see Appendix A) or until PTO2-T_latest(TSN) time has elapsed (see Section 3.3.1).

Rescue: If all of the following conditions are met:

* there are no outstanding TSN’s eligible for fast retransmission due to DupThresh or more miss indications

* there is no new data available for transmission and no data is outstanding on the association beyond the Fast Recovery Exit Point

* there are no outstanding TSNs eligible for Last Resort Retransmission

* the cumack has progressed since this entering of Fast Recovery

and there exist non-SACKed, non fast retransmitted TSNs, within the Fast Recovery Exit point, then for this entry of Fast Recovery, conditionally to that the CWND allows, we allow for fast retransmission of one packet of consecutive outstanding non fast retransmitted TSNs up to PMTU size, the highest TSN of which MUST be the highest outstanding TSN within the Fast Recovery Point. The bytes of a TSN which is retransmitted in this manner are not subtracted from the Flight size prior to this action be taken nor as a result of this action. If the miss indication count of the TSN subsequently reaches the DupThresh value, the bytes of the TSN shall be subtracted from the Flight size. Once acknowledged the

remaining contribution of this TSN in the Flight size (whether it be there counted once or twice at this point in time) is subtracted. A TSN which is retransmitted in this manner will be marked as ineligible for a subsequent fast retransmit(see considerations on Multiple Fast Retransmission operation in Section 3.3.1.3).

An implementation of the Rescue operation may be accomplished by maintain of an RescueRTX parameter as described for TCP in [RFC6675].

An SCTP implementation which implements the Unambiguous SACK feature of Appendix A may implement a more accurate calculation of the flightsize when performing Rescue operation. That is, instead of subtracting the contribution from the retransmitted TSN from the flightsize once the acknowledgement of the TSN arrives, the SCTP implement may distinguish where the acknowledgment is for the original TSN or for the retransmitted TSN and in case the acknowledgement is not for the retransmitted TSN, SCTP should delay the subtract of the bytes of the retransmitted TSN from the flightsize until either an acknowledgement of the retransmitted TSN is received (see Appendix A) or until PTO2-T_latest(TSN) time has elapsed (see Section 3.3.1).

DISCUSSION: [RFC4960] in addition to the HTNA algorithm demand for additional miss indication counting to be performed during Fast Recovery according to the following prescription (section 7.2.4 of [RFC4960]):

(#) If an endpoint is in Fast Recovery and a SACK arrives that advances the Cumulative TSN Ack Point, the miss indications are incremented for all TSNs reported missing in the SACK.

It is noted that under special circumstances then (#) makes SCTP Fast Recovery complete in situations where TCP Fast Recovery would only complete by virtue of the measure 3) or 4) of [RFC6675] and as such these measures are more critically demanded for TCP Fast Recovery operation than for the SCTP Fast Recovery operation. However as documented by (OPEN ISSUE: to be filled in) the Last Resort Retransmission operation and the Rescue operation also for SCTP significantly improve the Loss Recovery operation; the latency of the individual loss recovery operation as well as the ability of the operation to complete without resort to T3-timeout. Consequently this document prescribes for SCTP TLR to implement these procedures. Conversely even when the measures 3) and 4) of [RFC6675] are implemented, (#) gives benefits in terms of releasing flight size space allowing Fast Recovery to progress.

As the algorithm extension is limited by the existing congestion control algorithm of SCTP, these extensions of SCTP Fast Recovery do not compromise the TCP fairness of the SCTP Fast Recovery Operation.

3.2.1. Multi-Path Considerations

In multi-homed [RFC4960] SCTP, data that potentially will be subject to Fast Retransmission may be in flight on multiple paths. This (exception) situation in particular can occur as a result of a change of the data transfer path as a result of a switchback operation to a primary path. Here SACKs of data sent on one path (e.g., the new data transfer path) may result in generation of (potentially spurious) loss event reactions on a different path (the prior data transfer path). The [RFC4960] miss indication counting based on a common SACK clock is not changed in this document, nevertheless the protocol operation, here the operation of the Last Resort Retransmission and the Rescue operation in this situation, need to be specified.

The specification in this document is based on the following fundamental goals:

- an [RFC4960] SCTP implementation must appropriately react to loss events observed by means of miss indication counting, by performing appropriate adjustments of CWND and sstresh, an all paths where such loss events are observed.

- The observation of a loss event on one path should not for [RFC4960] SCTP MH impact the congestion control operation on a different path.

For the implementation of the Last Resort Retransmission and the Rescue operations for [RFC4960] MH SCTP then the following specifications are given:

- For a TSN to be eligible for Last Resort Retransmission a loss event MUST have been observed on the path on which this TSN is in flight.

- For a TSN to be eligible for the Rescue operation a loss event MUST have been observed on the path on which this TSN is in flight.

An implementation of the above may be accomplished by the implementation of a Fast Recovery state and Fast Recovery Exit point on a per path basis with the following particulars:

- o A path enters the Fast Recovery State based on loss event observation of TSNs in flight on the path.

- o When a loss event is observed on a path the Fast Recovery Exit point on the path is set to the highest TSN in flight of the path.

- o Fast Retransmission of TSNs in flight on the path terminates once the Fast Recovery Exit Point on the path has been reached (i.e., has been cumulative SACK'ed) at which point the Fast Recovery process on the path is terminated.

- o The eligibility of a TSN for the Last Resort Retransmission and the Rescue operation shall follow the prescriptions given above with adherence to the Fast Recovery Exit point set on the path on which the TSN is in flight.

The data retransmission process of data chunks in itself is prescribed to happen on the present data transfer path of the association regardless of which path the data chunks were in flight on when they became eligible for Fast Retransmission. This follows [RFC4960] and the preceding [CARO02].

With the above per path modelling of the Fast Recovery operation, SCTP may have multiple fast recovery exit points at any given time (though at most one per path) and the fast recovery operation may terminate at different times on the different paths. Further it is noted that a path may be in Fast Recovery even if no data is in flight on the path or even if the only data in flight on the path is beyond the Fast Recovery Exit Point of the path. The latter can occur in the very peculiar case where fast retransmission of data declared lost on the path happens on a different path as well as that the user performs a data directed data transfer on the path in question.

An SCTP implementation fulfilling the goals described above may also be achieved by other means than by maintain of a per path Fast Recovery Exit point. For example it might be achieved by maintain of a common association Fast Recovery Point spanning multiple paths, but still the implementation must ensure appropriate per destination address congestion control operation.

3.3. SCTP-TLR Description

3.3.1. Principles

The SCTP TLR function is based on the following principles.

3.3.1.1. Retransmission Timers Management

This document is specified as if there is a single retransmission timer per destination transport address, but implementations MAY have a retransmission timer for each DATA chunk.

This document specifies usage of new PTO timer for SCTP TLR. The document is specified as if the PTO timer functions are implemented by means of the existing retransmission timer of [RFC4960] SCTP, i.e., under certain conditions the retransmission-timer is activated with special PTO values rather than with the standard T3-timer value. The document is specified as if there is a single PTO timer per destination transport address, equivalently a single PTO timer per path. Implementations MAY choose to implement a PTO timer per DATA chunk.

For an outstanding TSN we define the time T_latest(TSN) to be the time that has elapsed since the TSN was last sent. When a TSN is first sent, or when it is retransmitted, T_latest(TSN)=0. An SCTP TLR implementation must be able to deduce this value for any outstanding TSN.

3.3.1.2. Timer driven entering of Fast Recovery

Timer driven entering of Fast Recovery in SCTP TLR is based on the following principles:

- o Maintain of a Tail Loss Probe Timer (PTO) which in certain situations (generally when retransmission is not performed) is running on a path. At any given time the value of the PTO timer is related to the lowest TSN in flight on the path. The PTO timer value used will depend on the situation:

  By default the following timer value is used:

  PTO1: PTO=MIN(RTO, 1.5*SRTT+MAX(RTTVAR, DELAY_ACK))

  Whereas the following value is used:

  PTO2: PTO=MIN(RTO, 1.5*SRTT+RTTVAR)

  when it is known that subsequent SACKs not acknowledging the TSN for which the PTO is running will be (or will have been) returned immediately. For more details see Section 3.3.2.

  By design the probe timer is kept lower or equal to the RTO, thereby aiming to prevent a potential unnecessary and damaging RTO, as well as generally larger than an anticipated RTT

thereby preventing that it kicks in prematurely. I.e., the timer only kicks in at a time where one would have expected to have received a SACK of the lowest TSN in flight were there no problems.

A minimal PTO value, PTO_MIN, is applied to the above formulas (particularly important for PTO2). I.e., the effective PTO1 = MAX(PTO_MIN, PTO1) and the effective PTO2 = MAX(PTO_MIN, PTO2). The suggested value of PTO_MIN is 10 msec. In the following when referring to PTO1 and PTO2 we refer to the effective PTO1 and PTO2 values.

For an SCTP implementation which performs RTT measurements during the association set-up, the PTO set on the path on which the first data chunk is sent shall be initialized from the RTT measured on the path during the association set-up. If no such RTT measurement is performed or is available on the particular path in question, the PTO shall be initialized as RTO_INIT.

- o PTO timer driven transmittal of Tail Loss Probe Packet: Once data is outstanding on a path and the PTO timer of the path kicks and no SACKs of any chunks with higher TSN number have arrived, a probe packet, denoted a Tail Loss Probe Packet (TLPP), is sent to probe for network responsiveness (i.e., for SACK of the TLPP) in order to potentially drive proactive entering of Fast Recovery.

  - * For a SCTP sender that supports the Immediate SACK feature, [RFC7053], the I-bit MUST be set on chunks sent in a TLPP packet.

- o PTO timer driven entering of Fast Recovery: Process is enforced when network responsiveness is proven (SACK of later sent data than lowest TSN in flight on the path is available) and (at least) PTO time has elapsed since transmittal of this lowest TSN in flight on the path.

Comment: The lowest outstanding TSN on an association may under special circumstances not be in flight on any path of the association. This can happen when the lowest outstanding TSN has been declared lost but the transmittal of the TSN is prevented due to congestion window limitations (e.g., during Fast Recovery). In this case, as well as generally for TSNs that are being retransmitted due to fast retransmission or T3-timeout, no PTO timer is running on the TSN. Conversely when the lowest outstanding TSN on a path is not subject to Fast Recovery or T3-Recovery, then this lowest outstanding TSN is also in flight on the path.

3.3.1.3. Fast-Recovery and Loss Detection

Fast Recovery and miss indication counting for the SCTP TLR function MUST embed the enhancements described in Section 3.2. In addition SCTP TLR implements the following loss detection during Fast Recovery:

- o If in Fast Recovery, then an outstanding TSN in flight on the path, with TSN lower that the Fast Recovery Exit Point on the path, is declared lost when the following conditions are satisfied:
  - * The TSN has not been fast retransmitted.
  - * T_latest(TSN) > PTO2.
  - * The TSN is lower than the highest outstanding SACK'ed TSN.

When declared lost by this procedure the TSN is subtracted from the flight size as well as it becomes eligible for fast retransmission as if it had been declared lost by reach of Dupthresh miss indication counts.

Such loss detection during SCTP TLR Fast Recovery shall at a minimum be done at receipt of SACK as well as at times where the possibility to transmit new data is being evaluated. An implementation maintaining PTO timers on a per data chunk basis may make further evaluation based on timer expiration.

Following [RFC4960] it is assumed that a data chunk should only be fast retransmitted once. I.e., subsequent retransmissions of the data chunk must proceed as T3-retransmission. An SCTP TLR implementation MAY possibly implement Multiple Fast Retransmission operation following the principles described in [CARO01] extended to include the Last Resort Retransmission and Rescue operations. Such however is not covered by the specification given here.

3.3.1.4. T3-Recovery

[RFC4960] does not explicitly specify for an T3-Recovery phase to be supported for SCTP, nor does [RFC4960] explicitly demand for that a data chunk which has been T3-retransmitted cannot undergo fast retransmission. It can be an advantage that a lost T3-retransmitted data chunk may be recovered by timely fast retransmission rather than by a subsequently, potentially back-off'ed T3-retransmission. For [RFC4960] MH SCTP, however, reliable implementation of such fast recovery of lost T3-retransmitted data is difficult to achieve given the usage of one common SACK clock as new data on one path may trick

spurious fast retransmission of data that has been/is being T3-retransmitted on a different path. Here it is important to emphasize that concurrent T3-retransmission and new data transmission on different paths is the standard operation of MH SCTP [RFC4960]. (Though implementations might possibly mitigate such effects by only sending new data after completion of the T3-retransmission operation as well as the implementation of SCTP-PF, [SCTP-PF], would further decrease the likelihood of such concurrent data transfer occurring.)

In this document we assume that an SCTP implementation follows either of the following implementation choices:

- o A data chunk which has underwent T3-retransmission cannot subsequently be subject to Fast Retransmission whether such entering of Fast Recovery be driven alone by miss indication counting or by the SCTP TLR mechanism. This implementation choice corresponds to implementing a T3-Recovery phase for SCTP equivalent with the RTO-recovery phase of TCP.

- o A data chunk, which has underwent T3-retransmission, will be eligible for subsequent Fast Retransmission if such is driven by miss indication counts from SACKs of new data chunks sent after all data outstanding for T3-retransmission have been sent and the new data is sent on the same path as the T3-retransmission data.

One implementation choice may be to follow the first implementation choice for SCTP MH and the second implementation choice for SCTP SH. Regardless of this implementation choice then in SCTP TLR a data chunk that has been subject to T3-retransmission SHOULD NOT by subject to the timer driven entering of Fast Recovery specified below. The motivation for this choice is that the SRTT may not be appropriately refreshed during the T3-retransmission process. OPEN ISSUE/TO DO: Ideally the PTO timer used after the exit of the T3-recovery phase should be updated based on a fresh RTT measurement. E.g., from the last acknowledged TSN. If no new SRTT calculation is made based on a scheduled RTT measurement, then the PTO timer values could be made sure to be appropriately adjusted, if necessary, by a last measured RTT by 1,5*SRTT + RTTVAR --> MAX(1*5 RTT, 1,5*SRTT + RTTVAR).

3.3.2. SCTP - TLR Statemachine

The SCTP Tail Loss Recovery function defines 3 states: The SCTP TLR OPEN state, the SCTP TLR PROBE WAIT state and the SCTP TLR DELAY WAIT state. At any given time the SCTP transmission logic for the lowest outstanding TSN on a path will be in one of these 3 states or the TSN is sought being recovered by means of Fast Recovery or T3-Recovery.

Figure 1 illustrates the states and the state transitions.

(to be inserted)

Figure 1, Enhanced Loss Recovery State Machine Diagram

In the following we describe the states and the actions taken.

3.3.2.1. SCTP TLR OPEN STATE

This is the state the SCTP transmission logic is in on any path when no TSN is outstanding on the association as well as it is the state when SCTP sends the first data on a path after idle/no TSN outstanding. It also more generally is the state the transmission logic is in when there are no gaps in the SACK scoreboard beyond the lowest outstanding TSN on the path.

In this state SCTP is not performing Fast Recovery nor T3-Recovery on the lowest TSN outstanding on the path and no SACKs of any chunks with higher TSN number have arrived. In this state, when SCTP has outstanding data on the path, a PTO timer is running relative to the lowest TSN outstanding on the path.

The PTO set on a (new) lowest outstanding TSN on the path in this state will follow PTO1 when less than 2 packets are outstanding beyond the TSN at the time when the timer is set and follow PTO2 when 2 or more packets are outstanding beyond the TSN when the PTO timer is set or when the Immediate SACK feature is known to be supported by both sender and receiver (see Section 4) and the I-bit has been set on the TSN or on an outstanding TSN of higher number.

In the OPEN state the following may happen:

- o A SACK commutatively acknowledging the lowest outstanding TSN and resulting in no gaps in the SACK scoreboard may arrive. In this case the state remains in OPEN state. If there still is outstanding data on the path, the PTO timer is set on the new lowest outstanding TSN. The PTO timer value set will be the value PTO - T_latest(TSN) where the PTO value is calculated either from PTO1 or PTO2 according to the evaluation criteria given above.

- o A SACK with gap(s) may arrive, thus proving network responsiveness while still not cumulatively acknowledging all lower (than the SACK’ed gap) outstanding TSNs on the path. The SACK may or may not move the cumulative ACK point. This indicates that either

packets are being re-ordered or the (new) lowest outstanding TSN on the path has been lost.

* If the SACK makes the miss indication count on the (new) lowest outstanding TSN reach Dupthresh the SCTP OPEN state is terminated and Fast Recovery is started.

* If Dupthresh miss indication count is not reached on the (new) lowest outstanding TSN, the state will now transit to SCTP TLR DELAY WAIT state for potential entering of SCTP TLR driven Fast Recovery if the PTO timer kicks prior to the (new) lowest outstanding TSN has been acknowledged or for potential later entering of Fast Recovery by reach of Dupthresh miss indication counts. When transiting to SCTP TLR DELAY WAIT the PTO timer relative to the (new) lowest outstanding TSN is reset to PTO2 - T_latest(TSN). In case PTO2 - T_latest(TSN) <= 0, the DELAY WAIT state is immediately terminated, the packet containing the lowest outstanding TSN is declared lost, and Fast Recovery is started.

o The PTO timer relative to the lowest outstanding TSN may kick, in which case SCTP TLR will send a TLPP, reset the PTO timer relative to the lowest outstanding TSN to a T3 timer and transit to SCTP TLR PROBE WAIT state to await either the kick of the T3 relative to the lowest outstanding TSN (network is persistently unresponsive) or proof of network responsiveness and potential entering of SCTP TLR driven Fast Recovery unless the network responsiveness proof comes in form of cumulative acknowledgement of the TSN. The T3-value set relative to the lowest outstanding TSN when sending the TLPP probe and entering this state shall be:

  * MAX(PTO1, RTO - T_latest(TSN))), when receiver side support for Immediate SACK has not been confirmed for the association, see Section 4.

  * MAX(PTO2, RTO - T_latest(TSN)), when receiver side support for Immediate SACK has been confirmed for the association, see Section 4, and the SCTP sender itself deploys the Immediate SACK feature.

  For further details on the TLPP transmission see Section 3.3.3.

3.3.2.2. SCTP TLR PROBE WAIT STATE

In this state the lowest outstanding TSN has remained unSACK’ed for more than PTO time and no indication (no SACK of higher outstanding TSNs have been received) thus resulting in the transmittal of a TLPP to probe for the network responsiveness.

The T3-value set relative to the lowest outstanding TSN when sending the TLPP probe and entering this state is:

- o MAX(PTO1, RTO - T_latest(TSN))), when receiver side support for Immediate SACK has not been confirmed for the association, see Section 4.

- o MAX(PTO2, RTO - T_latest(TSN)), when receiver side support for Immediate SACK has been confirmed for the association, see Section 4, and the SCTP sender itself deploys the Immediate SACK feature.

For further details on the TLPP transmission see Section 3.3.3. Observe that in some special cases no TLPP is sent even if this state is entered and conceptually is handled as if a TLPP has been sent.

In the PROBE WAIT state the following may happen:

- o SACKs may arrive that makes the miss indication count on the lowest outstanding TSN/lowest TSN in flight reach Dupthresh in which case the PROBE WAIT state is terminated and Fast Recovery is started.

- o A SACK cumulatively acknowledging all holes including the lowest outstanding TSN may bring the SCTP TLR STM state back to SCTP TLR OPEN state. In this case a new PTO timer will be started on the new lowest outstanding TSN following the PTO timer setting in the SCTP TLR OPEN state. In this situation "PTO restart principles" (i.e., yielding PTO-T_latest(TSN)) shall not be deployed. Spurious entering of PROBE WAIT state can happen if the PTO is too short, in such a situation it would not be prudent to deploy PTO restart principles when returning to OPEN state. OPEN ISSUE: Possibly PTO restart principles shall be refrained from until new RTT measurements are available.

- o A SACK may arrive for a higher outstanding TSN with lowest outstanding TSN on the path remaining unSACK’ed. This will result in declaration of the packet of the lowest outstanding TSN as lost and will make SCTP enter Fast Recovery.

- o A SACK may arrive that acknowledges the lowest outstanding TSN, but also data of higher TSN than the new lowest outstanding TSN are acknowledged in the SACK. In this case there is indication that either packet re-ordering has occurred or the new lowest outstanding TSN has been lost. The state will now transit to SCTP TLR DELAY WAIT state for potential entering of SCTP TLR driven Fast Recovery if the PTO timer kicks prior to the new lowest outstanding TSN has been acknowledged. The PTO timer set on the

new lowest outstanding TSN will be PTO2 - T_latest(TSN). In case PTO2 - T_latest(TSN) <= 0, the DELAY WAIT state is immediately terminated, the packet containing the lowest outstanding TSN is declared lost, and Fast Recovery is started.

- o The T3-timer may kick. In this case the PROBE WAIT state will be terminated and T3-recovery will start on non-SACK'ed outstanding data.

3.3.2.3. SCTP TLR DELAY WAIT STATE

In this state network responsiveness has been received (in form of a SACK of higher TSN than the lowest outstanding TSN) and the PTO timer relative to the lowest outstanding TSN is running for potential entering of SCTP TLR driven Fast Recovery.

The PTO set on a new lowest outstanding TSN in this state will be according to PTO2 in form of PTO2-T_latest(TSN).

In the DELAY WAIT state the following may happen:

- o SACKs may arrive that will make the miss indication count on the lowest TSN in flight reach Dupthresh, the DELAY WAIT state is terminated and SCTP enters Fast Recovery.

- o The PTO timer relative to the lowest outstanding TSN may kick. This will result in declaration of packet of the lowest outstanding TSN as lost and will make SCTP enter Fast Recovery.

- o A SACK cumulatively acknowledging all holes including the lowest outstanding TSN may arrive and bring the SCTP TLR STM state back to SCTP TLR OPEN state and the PTO timer will be restarted on the new lowest outstanding TSN. The PTO timer value set will be the value PTO - T_latest(TSN) where the PTO value is calculated either from PTO1 or PTO2 according to the evaluation criteria given for the OPEN state.

- o A SACK may arrive that acknowledges the lowest outstanding TSN, but also data of higher TSN than the new lowest outstanding TSN are acknowledged in the SACK. In this case there is indication that either packet re-ordering has occurred or the new lowest outstanding TSN has been lost. The state will remain in SCTP TLR DELAY WAIT state for potential entering of SCTP TLR driven Fast Recovery if the PTO timer kicks prior to the new lowest outstanding TSN has been acknowledged. The PTO timer set on the new lowest outstanding TSN will be PTO2 - T_latest(TSN). In case PTO2 - T_latest(TSN) <= 0, the DELAY WAIT state is terminated, the

packet containing the lowest outstanding TSN is declared lost and Fast Recovery is started.

- o A SACK may arrive that does not acknowledge the lowest outstanding TSN and still do not make the miss indication count reach the Dupthresh value. In this situation no changes are done to the PTO timer running and the state will remain in SCTP TLR DELAY WAIT state for potential entering of SCTP TLR driven Fast Recovery if the PTO timer kicks prior to the lowest outstanding TSN has been acknowledged.

3.3.2.4. Exit of Loss Recovery

After exit of Fast Recovery or completion of T3-retransmission then if data is outstanding a PTO timer is started relative to the lowest outstanding TSN on the path and the state transits to either SCTP TLR OPEN state or to SCTP TLR DELAY Wait state depending on the status of the SACK scoreboard (i.e., do gaps exist or not). The PTO timer set will follow the rules described above. PTO-restart principles shall not be deployed in this situation as fresh RTT measurements might not be available. OPEN ISSUE: Possibly PTO restart principles shall be refrained from until new RTT measurements are available.

3.3.2.5. RTO-Restart Principles for the T3-timer

When the lowest TSN in flight on a path is undergoing Fast Recovery or T3-retransmission a T3-timer is running on the path (relative to this lowest TSN in flight). For SCTP TLR the RTO-restart principles as of [HURTIG] SHOULD unconditionally be applied to the T3-timer. Thus the T3-timer set on a path in this case SHOULD be the value RTO-T_latest(TSN) relative to the lowest TSN in flight on the path.

3.3.3. TLPP Transmission Rules

The transmission of a Tail Loss Probe Packet (TLPP), done just prior to entering the SCTP TLR PROBE WAIT state from SCTP OPEN, is governed by the following details:

- o TLPP of new data is always preferred if such is available for transmission. If such exists, the TLPP sent is chosen as the lowest unsent TSNs that fit into one packet

- o Alternatively if no new data is available for transmission, either due to application or receiver side limitations, the presently outstanding packet with highest TSN number is chosen as the TLPP.

- o TLPP of retransmission data counts twice in the in-flight until acknowledged or detected as lost.

- o The transmittal of a TLPP of sub-PMTU size is not blocked by Nagle-like bundling.

The highest (new) outstanding TSN is chosen for probing in order to best possibly interface with standard Fast Recovery, i.e., to create a loss pattern situation that corresponds best possibly with how Fast Recovery algorithm retransmits, and is invoked to retransmit, lost packets.

TLPP Transmission conditions:

A TLPP is not sent unconditionally when SCTP enters PROBE WAIT state on a path.

No explicit limit is applied to the number of TLPP probe packets (i.e., the number of unacknowledged packets sent as TLPP) that may be outstanding at any given time but the number of such will in most situations be effectively limited to a very few (very often only one) by the following rules based on latency and congestion control principles; Generally a TLPP will not be allowed to breach the CWND more than once per RTT and further a TLPP is omitted to be sent if an already outstanding packet is considered to serve "good enough" from a network probing perspective. In addition special considerations are given for the transmittal of a TLPP consisting of retransmission data to ease loss masking detection (see Section 3.3.4). It is further noted that the frequency of TLPP transmittal is limited by how often a transition can happen out of and back into the PROBE WAIT state.

The conditional transmission of a TLPP is specified as follows:

- o If the highest outstanding TSN has been sent only a little while ago, this TSN effectively serves as a probe and no TLPP need to be send. This condition aims to prevent unnecessary retransmission of just sent data and unnecessary transmittal of small sub-PMTU packets of new data. The exact condition to apply is:

  * If T_Latest(highTSN) < gamma * SRTT

  then no TLPP is sent. gamma = 1/2 is recommended. A special condition arise when little data is outstanding and the SACK of the outstanding data may be lost by a single loss of SACK. In this case the transmittal of a TLPP packet will make the SACK return be robust toward single loss of SACK. For added robustness to SACK return an SCTP TLR implementation MAY disregard the above condition if only 2 packets are outstanding.

- o If no TLPP is outstanding, a probe is sent unconditionally of CWND.

- o If a TLPP is outstanding, a probe is sent conditionally to that there is room in CWND. Otherwise no TLPP is sent. I.e., the CWND is not breached when a TLPP is outstanding.

- o If no new data exists, a probe of retransmission data is sent conditional to whether a TLPP of retransmission data is already outstanding. I.e.,:

  - * If no TLPP of retransmission data is outstanding, send TLPP consisting of highest outstanding TSN.

  - * If a TLPP of retransmission data is outstanding, no TLPP is sent.

The above rules on probes of retransmission data are defined to ease the detection of TLPP recovered losses by the algorithm described in Section 3.3.4.

3.3.3.1. Multi-Path Considerations for TLPP Transmission

In multi-homed [RFC4960] SCTP, multiple paths may have a PTO timer running on data in flight. E.g., two paths may be in SCTP OPEN state and SCTP will have two PTO timers running, each relative to the lowest outstanding TSN on the respective path. This (exception) situation in particular can occur as a result of a change of the data transfer path as a result of a switchback operation to a primary path. The handling of TLPP transmission for SCTP MH is described in the following. The underlying philosophy of the solution is, as far as possible, to have the SCTP TLR probing mechanism be undertaken on, and by, the data transfer path. Thus best possibly avoiding conflicts that may arise due to concurrent data transfers on multiple paths. As follows:

- o When the PTO timer kicks on a path in SCTP OPEN state and the TLPP selected by the rules above consists of new data, then if the path is the present data transfer path of the association the TLPP will be sent and in this case the TLPP is sent on the data transfer path of the association. When in this situation the path is not the present data transfer path of the association, then

  - * if there is no outstanding data on the present data transfer path, the TLPP of new data is sent there.

  - * if there is outstanding data on the data transfer path, the TLPP is not sent. Instead the potential transmittal of a TLPP

is deferred to be driven by a later kick of the PTO timer on the data transfer path.

The first situation that data is available for transmittal on the data transfer path but has not been sent, is an unlikely situation, but it might possibly occur in some implementations.

o When the PTO timer kicks on a path in SCTP OPEN state and the TLPP selected by the rules above consist of retransmission of the presently highest outstanding TSNs on the association, then if and only if these TSNs are outstanding on the path in question is the TLPP allowed to be sent. The following guidelines are given for the path selection for the TLPP:

  * An SCTP implementation which does not implement the Unambiguous SACK feature of Appendix A should send the TLPP on the path on which the TNSs are presently outstanding (i.e., on the path on which the PTO kicked).

  * An SCTP implementation which implements the Unambiguous SACK feature of Appendix A may send the TLPP on the data transfer path of the association.

  The reason a TLPP of retransmitted data in the first case above is sent on the path on which the data was first sent, even if this path is not the present data transfer path (special corner case with change of data transfer path or destination adders directed data transfer), is that the TLPP Loss Mask Detection mechanism, see Section 3.3.4 could not infer on which path to perform a congestion window reduction if the TLPP and original data is sent on different paths. An SCTP implementation which implements the Unambiguous SACK feature of Appendix A can better distinguish the SACK of the original TSN and the retransmitted TSN and can therefore operate differently. The choice of sending the TLPP on the data transfer path may be motivated by that the Fast Recovery procedure, which the SACK of the TLPP may result in, would use the data transfer path. On the other hand then differences in the RTT on the different paths may make it suboptimal to send the TLPP on the data transfer path as well as it can give rise to potential uncertainty in the TLPP Loss Recovery Mask detection and reaction process (see Section 3.3.4).

It is emphasized that the deferral of the transmission of a TLPP does not prevent entering of the PROBE WAIT state on the path where the PTO kicked.

3.3.4. Masking of TLPP Recovered Losses

If a single SCTP packet is lost, there is a risk that the TLPP packet itself might repair the loss if that particular lost packet is used as probe. The masking problem is only present if the TLPP is based on retransmission data. The TLPP might mask the loss and thus interfere with the congestion control principle that requires for CWND halving when a loss is detected.

At present the solution in this document operates with the algorithm defined for this purpose in [DUKKIPATI01] with adjustment to SCTP to rely on the D-SACK (duplicate TSN received) information available from SCTP SACK or alternatively to the information available from the Unambiguous SACK information of Appendix A. The solution operates with a conceptual TLPP Retransmission Episode. As follows:

- o Once a TLPP packet consisting of retransmission data is sent a TLPP Retransmission Episode is started.

- o A TLPP Retransmission Episode is abruptly terminated if Fast Recovery or T3-Recovery is entered.

- o For an SCTP implementation which does not implement the Unambiguous SACK feature of Appendix A, as well as for an SCTP association where the Unambiguous SACK feature of Appendix A is not in use, the TLPP Retransmission Episode terminates when an incoming SACK cumulatively acknowledges a sequence number higher than the sequence number of the TLPP probe with retransmission data. If at this time in stage the number of times the TLPP TSN has been received, according to the D-SACK information received, is lower than the number of times the TLPP TSN has been sent, CWND halving is done on the unique path on which the retransmission TLPP TSN has been sent. Further at this stage in time the contribution from the TSN is subtracted from the flight size in accordance to the number of times the TSN has been sent.

- o For an SCTP implementation which implements the Unambiguous SACK feature of Appendix A the following actions are taken at the time of acknowledgement of the TSN used as TLPP:

  - * If the TLPP TSN is first cumulatively acknowledged in a SACK with CUMACK TSN = TLPP TSN and with no SACK (or CUMACK) of higher TSNs, then from the Unambiguous SACK information SCTP sender can classify to be in the following cases:

    - + The original TSN has not (yet) been received, the retransmission TSN (the TLPP) has been received.

- In this case the original TSN is judged as lost, CWND halving is performed on the path on which the original TSN was sent and the sent TSNs are subtracted from the flight size(s). This concludes the TLPP Retransmission Episode.

+ Both the original transmission as well as the retransmission (the TLPP) have been received.

  - In this case the sent TSNs are subtracted from the flight size(s). This concludes the TLPP Retransmission Episode.

+ The original TSN has been received, the retransmission TSN (the TLPP) has not yet been received:

  - In this case a special timer is started with value PTO-T_latest(TSN)and the bytes of the retransmitted TSN (the TLPP) remains in the flightsize of the path on which it was sent until either of the following happens - whichever happens first:

    o Unambiguous SACK of the TSN is received in which case the TSN is subtracted from the flightsize and the timer is stopped. This concludes the TLPP Retransmission Episode.

    o A SACK of a higher TSN than the TLPP arrives with unambiguous SACK information indicating that the TLPP has not been received. Now marking is made on the path so that, if when the timer kicks, the TSN has still not been acknowledged, the TSN is judged as lost, CWND halving is done and the TSN is subtracted from the flightsize. This then concludes the TLPP Retransmission Episode.

    o The timer kicks, the TSN is subtracted from the flightsize (but no CWND halving is done). This concludes the TLPP Retransmission Episode.

* If the TLPP TSN is first cumulatively acknowledged in a SACK with highest SACK’ed (or CUMACK’ed) TSN > TLPP TSN, then from the Unambiguous SACK information SCTP sender can classify the same cases as above and take corresponding actions. One additional situation can arise in this situation:

  + Only one of the transmissions of the TSN has been received, but no clear Unambiguous SACK indication of which that was received is available from the SACK. This uncertainty can

only result from situations where SACKs are lost, potentially in combination with that more data chunks than the TSN it self were outstanding at the time when the TLPP was sent and some of this data arrived later at the receiver than the original TSN or the TLPP.

- In this case the original TSN is judged as having been received and it is subtracted on the flightsize of the path on which it was sent. The timer PTO-T_latest(TSN) is set and handling of potential CWND reduction caused by loss of the TLPP is handled following the principles described above.

DISCUSSION of Unambiguous SACK Case Handling: CWND halving is not prescribed to be done for a potential lost retransmitted TSN used as TLPP in all cases above as there is no guarantee that a SACK confirming a potential arrival of the retransmitted TSN will arrive in time (i.e., this SACK may be lost). CWND halving is done if SACK of a higher TSN number than the TLPP number has arrived, PTO time has elapsed since the transmittal of the TLPP and the TLPP in it self cannot be determined to be received from the Unambiguous SACK information.

3.3.5. Elimination of unnecesary DELAY-ACK delays

The negative impact of DELAY_ACK on the loss recovery delay is partially mitigated by setting of the I-bit on TLPP.

OPEN ISSUES:

o It is to be determined if the Immediate SACK feature shall be relied on more aggressively. Possible options are:

  * Immediate SACK flag to be set on all retransmitted TSNs.

  * Immediate SACK flag to be set on all TSNs that are sent where the transmittal of an immediate following subsequent packet cannot be foreseen. This effectively would result in that the I-bit is set on a sent TSN whenever either of the following is true:

    + no more chunks can be sent right after this chunk due to CWND limitations.

    + no more chunks can be sent right after this due to RCV window limitations

+ no more chunks can be sent right after this as no more chunks are available in the SND buffer.

+ no more chunks can be sent right after this due to Nagle. (May depend on the exact Nagle-like implementation).

For the second choice it would be relevant to use PTO1 setting for the PTO timer on all TSNs sent with the I-bit set, when the receiver is known to support the Immediate SACK feature. The downside of this choice is that it very severely limits the effectiveness of the DELAY_ACK feature.

o Ideally the PTO timer relative to the lowest outstanding TSN should be adjusted to follow PTO2 when a subsequent packet is transmitted. The downside of this choice is the implementation impacts of such detailed - potentially per packet transmission - logic. To be elaborated further.

4. Confirmation of support for Immediate SACK

Confirmation of receiver support of the Immediate SACK function, [RFC7053] is established by an SCTP TLR sender by the following means:

o In case the data chunk of [RFC4960] is in use on the association, confirmation of [RFC7053] support by the SCTP receiver is assumed if SCTP TLR sender receives a data chunk with the I-bit flag set.

o [TO DE CONFIRMED:] In case the I-data chunk of [SCTP-IDATA] is in use on the association, SCTP sender can by [SCTP-IDATA] assume that SCTP receiver supports [RFC7053].

5. Socket API Considerations

This section will describe how the socket API defined in [RFC6458] is extended to provide a way for the application to control the retransmission algorithms in operation in the SCTP layer.

Socket option for control of the features is yet to be defined.

Please note that this section is informational only.

6. Security Considerations

There are no new security considerations introduced by the functions defined in this document.

7. Acknowledgements

The author acknowledges Henrik Jensen for his very significant contribution for the definition of, the implementation of and the experiments with function.

The work heavily draws on prior art work done for TCP, [DUKKIPATI01] in particular. The contributors of that work should be credited for many of the ideas put forward here for SCTP.

8. IANA Considerations

This document does not create any new registries or modify the rules for any existing registries managed by IANA.

9. Discussion and Evaluation of function

Experiments in progress. Details to be filled in.

Right now we use this section to retain a number of issues that are to further elaborated on:

- o A significant number of spurious TLR probes have been observed in tests. It is to be determined if this is a fact of the function or whether it may be improved with adjustment of the PTO timer calculations.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <http://www.rfc-editor.org/info/rfc5061>.

[RFC5062] Stewart, R., Tuexen, M., and G. Camarillo, "Security Attacks Found Against the Stream Control Transmission Protocol (SCTP) and Current Countermeasures", RFC 5062, DOI 10.17487/RFC5062, September 2007, <http://www.rfc-editor.org/info/rfc5062>.

[RFC7053] Tuexen, M., Ruengeler, I., and R. Stewart, "SACK-IMMEDIATELY Extension for the Stream Control Transmission Protocol", RFC 7053, DOI 10.17487/RFC7053, November 2013, <http://www.rfc-editor.org/info/rfc7053>.

[SCTP-IDATA] R. Stewart et al, , "Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol draft-ietf-tsvwg-sctp-ndata-04.txt", IETF Work In Progress , 07 2015.

10.2. Informative References

[CARO01] A. Caro et al, , "Retransmission Policies with Transport Layer Multihoming", ICON , 2003.

[CARO02] A. Caro et al, , "Retransmission Schemes for End-to-end Failover with Transport Layer Multihoming", GLOBECOM , 11 2004.

[CMT-SCTP] Amer et al., P., "Load Sharing for the Stream Control Transmission Protocol (SCTP) draft-tuexen-tsvwg-sctp-multipath-10.txt", IETF Work In Progress , 5 2015.

[DUKKIPATI01] Dukkipati, N., Cardwell, N., Cheng, Y., and M. Mathis, "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail", Work Expired , 2 2013.

[DUKKIPATI02] Dukkipati, N., Mathis, M., Cheng, Y., and M. Ghobadi, "Proportional Rate Reduction for TCP", Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement , 11 2011.

[HURTIG] P. Hurtig et al., , "TCP and SCTP RTO Restart, draft-ietf-tcpm-rtorestart-08", IETF Work In Progress , 3 2015.

[MATHIS] Mathis, M., "FACK", ACM SIGCOMM Computer Communication Review 26,4, 10 1996.

[Rajiullah]
M. Rajiullah et al., , "An Evaluation of Tail Loss Recovery Mechanisms for TCP", ACM SIGCOMM Computer Communication Review 45,1, 1 2015.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <http://www.rfc-editor.org/info/rfc3758>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC5827] Allman, M., Avrachenkov, K., Ayesta, U., Blanton, J., and P. Hurtig, "Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)", RFC 5827, DOI 10.17487/RFC5827, May 2010, <http://www.rfc-editor.org/info/rfc5827>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <http://www.rfc-editor.org/info/rfc6458>.

[RFC6675] Blanton, E., Allman, M., Wang, L., Jarvinen, I., Kojo, M., and Y. Nishida, "A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP", RFC 6675, DOI 10.17487/RFC6675, August 2012, <http://www.rfc-editor.org/info/rfc6675>.

[SCTP-PF] Y. Nishida et al, , "SCTP-PF: Quick Failover Algorithm in SCTP, draft-ietf-tsvwg-sctp-failover-13.txt", IETF Work In Progress , 09 2015.

[zimmermann01]
Zimmermann, A., "CUBIC for Fast Long-Distance Networks, draft-ietf-tcpm-cubic-00", IETF Work In Progress , 6 2015.

[zimmermann02]
Zimmermann, A., "The TCP Echo and TCP Echo Reply Option, draft-zimmermann-tcpm-echo-option-00", IETF Work In Progress , 6 2015.

[zimmermann03]
Zimmermann, A., "Using the TCP Echo Option for Spurious Retransmission Detection, draft-zimmermann-tcpm-spurious-rxmit-00", IETF Work In Progress , 7 2015.

Appendix A. Unambuiguous SACK

When receiving a SACK of a TSN it is not possible to unambiguously determine if the receiver hereby acknowledges the first transmission of the TSN or possible subsequent retransmissions of the TSN, when such multiple transmissions of the same TSN have been made. The duplicate TSN information in the SCTP SACK chunk does help to provide information about how many times the same TSN has been received at the received side, but still it is not possible to unequivocally link the SACK information to the different transmissions of the same TSN. An additional source of ambiguity comes from the fact that packets may be duplicated in the network.

Unambiguous SACK information is generally beneficial for many SCTP protocol aspects, e.g., for improved RTT measurements, for more accurate loss detection, maintain of flightsize and congestion control operation.

Providing full accurate SACK information from receiver to sender side requires a reliable (and ordered) SACK feedback channel thus overcoming the information gap that may arise from loss (or from re-ordering) of SACKs. The establishment of such a reliable feedback Chanel is not proposed but the proposal implements measures that allow for some robustness towards information loss due to SACK loss.

NOTE for AUTHORS: The solution is independent from a potential split of the SACK TSN Gap information in SACK and NR-SACK gaps respectively following [CMT-SCTP].

A.1. TSN Retransmission ID in Data Chunk Header

It is a prerequisite that the SCTP association deploy, and has negotiated usage of, the new I-data chunk of [SCTP-IDATA].

We define a new 4-bit Retransmission ID (RTX ID) in the I-data Chunk header. The 4 bits consume 4 bits of the new reserved 16-bit filed of the I-data chunk header. See Figure 1.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 64   |  Res  |I|U|B|E|           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              TSN                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Stream Identifier      |   Reserved            | RTX-ID|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Message Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Payload Protocol Identifier / Fragment Sequence Number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                            User Data                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: RTX-ID in I-DATA chunk format

A.1.1. Sender side behaviour

New data MUST be sent with RTX-ID =0. Whenever SCTP retransmits a data chunk it SHOULD step up the RTX ID. The highest RXT ID = 15 is used for all retransmissions of the same TSN beyond the 15-th retransmission or when the RTX ID last used fort his TSN is 15. An SCTP sender MAY step the RTX ID up with more than one count when retransmitting a TSNs in order to have all TSNs within the SCTP packet use the one and the same RTX ID.

A.1.2. Receiver side behaviour

An SCTP receiver supporting this feature MUST process the RTX ID for all received TSNs in accordance with the prescriptions for Unambiguous SACK return below.

A.2. Unambuiguous SACK Chunk

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = x    |Chunk  Flags   |      Chunk Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Cumulative TSN RTX (CUMACK TSN)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Advertised Receiver Window Credit (a_rwnd)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Number of Gap Ack Blocks = N  |  Reserved (future NR-SACK ?)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| NewlyCACK RTX ID Blocks = N   |   CACK Dupl TSN Blocks = N    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| NewlySACK RTX ID Blocks = N   |   SACK Dupl TSN Blocks = N    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Number of RTX SACK Blocks = N |  Reserved                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Highest CUMACK 'ed TSN received duplicated                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Gap Ack Block #1 Start       |   Gap Ack Block #1 End        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                                                               /
\    format to be changed to cover more than 16-bits ?          \
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Gap Ack Block #N Start      |  Gap Ack Block #N End         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                                                               /
\      New Blocks in order set above ... to be filled in        \
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Unambuiguous SACK chunk format

Newly CACK RTX ID block:

This block provides information on the newly acknowledged TSNs that were cumulatively acked in this SACK and for which the following hold:

* The TSN is newly acked in this SACK. I.e., the TSN has not been received before (or if it has been received before it was since reneged).

* The newly acknowledged TSN was received with RTX ID different from zero.

The RTX ID received with the TSN is returned in this block. The information returned in a CACK RTX ID block is a consecutive range of TSN fulfilling the above for which identical RTX ID has been received. Proposed format is off-set from CUMACK TSN (lower than CUMACK TSN), length of range and RTX ID.

Newly SACK RTX ID block:

This block provides information on the newly acknowledged TSNs that were selectively acknowledged in this SACK and for which the following hold:

* The TSN is newly acked in this SACK. I.e., the TSN has not been received before (or if it has been received before, it was since reneged).

* The newly acknowledged TSN was received with RTX ID different from zero.

The RTX ID received with the TSN is returned in this block. The information returned in a SACK RTX ID block is a consecutive range of TSN fulfilling the above for which identical RTX ID has been received. Proposed format is off-set from CUMACK TSN (higher than CUMACK TSN), length of range and RTX ID - OR alternatively format of present SACK blocks with off set bounded by 16-bit to CUMACK TSN.

Newly CACK Dupl TSN block:

This block provides information on the TSNs received since last returned SACK for which following hold:

* The TSN is lower than or equal to the CUMACK TSN.

* The TSN is a duplicate. Meaning that a data chunk with same TSN, but possibly different RTX ID, has been received.

The RTX ID received with the TSN is returned in this block. The information returned in a CACK Dupl TSN block is a consecutive range of TSN fulfilling the above for which identical RTX ID has been received. Proposed format is off-set from CUMACK TSN (lower than CUMACK TSN), length of range and RTX ID. The RTX ID may be zero.

Newly SACK Dupl TSN block:

This block provide information on the TSNs received since last returned SACK for which the following hold:

* The TSN is higher than the CUMACK TSN.

* The TSN is a duplicate. Meaning that a data chunk with same TSN, but possibly different RTX ID, has been received.

The RTX ID received with the TSN is returned in this block. The information returned in a SACK Dupl TSN block is a consecutive range of TSN fulfilling the above for which identical RTX ID has been received. Proposed format is off-set from CUMACK TSN (higher than CUMACK TSN), length of range and RTX ID - OR - format of present SAC blocks with off set bounded by 16-bit to CUMACK TSN. The RTX ID may be zero.

Together with the existing SACK information, the Newly CACK/SACK RTX ID and the CACK/SACK Dupl TSN blocks provide unambiguous SACK information for all received TSNs differentiating on the RTX ID received with the TSN. The information may be partially lost from the receiver to the sender if a SACK is lost. The RTX SACK Block and the Highest CUMACK Received Duplicated information is returned in order to provide means to recover part of the information that can be lost when a SACK is lost.

RTX SACK block:

This block provides information on the TSNs for which the following hold:

* The TSN has been received and has been selectively acked in prior SACKs (OPEN: alternatively in SACKs including this one).

* The TSN is higher than the CUMACK TSN.

* The TSN has been received only with RTX IDs different from zero.

The information returned in an RTX block is a consecutive range of TSN fulfilling the above. Proposed format is off-set from CUMACK TSN (higher than CUMACK TSN) and length of range - OR - format of present SACK blocks with off set - bounded by 16-bit to CUMACK TSN.

Highest CUMACK’ed TSN received Duplicated:

Here the highest TSNs that fulfill the following condition is inserted:

* The TSN has been received duplicated

* The TSN is lower than or equal to the CUMACK TSN.

When no duplicates have been seen or when no duplicates have been seen in last 2^31 window of TSNs that have been cumulatively acknowledged, CUMACK TSN +1 is returned.

By means of the RTX SACK block an SCTP sender may recover the information that a SACK'ed TSN does not represent the original TSN first sent. I.e., the TSN sent with RTX ID = 0.

By means of the "Highest CUMACK'ed TSN received Duplicated" an SCTP receiver may recover the information that more than one incarnation of a TSN has been received when the SACK, which cumulatively acknowledged the arrival of the different incarnations of the TSN, in it self was lost. The particular example of special interest is the case where the one and the same SACK would contain information on receipt of both the original TSN and a spurious retransmission of the TSN. Such can happen in scenarios where DELAY_ACK handling at the receiver side delays the return of SACK information and a SACK is lost, even if the original data and the spurious retransmission data was sent with reasonable spacing in time.

A.2.1. Receiver side behaviour

The RTX SACK Block and the Highest CUMACK information to be returned in SACKs demand for an SCTP receiver to keep track (state) of the following information on a per association basis:

o A list (or ranges) of TSNs that have been SACK'ed, but not yet cumulatively acknowledged and for which RTX ID = 0 has not been seen. It is noted that the TSN data chunk itself may have been delivered to the application.

o The highest TSN lower than CUMACK TSN for which a duplicate has been received.

A.3. Unambuigous SACK return

Whenever Unambiguous SACKs are in use on an association and SCTP receives a valid data chunk with RTX-ID different from zero it shall not delay the return of the Unambiguous SACK. Otherwise Unambiguous SACKs are returned at any time when an [RFC4960] implementation would return a SACK.

A window opener MUST include Unambiguous SACK information.

A.4. Negotiation

An SCTP receiver MUST NOT send an Unambiguous SACK chunk unless both peers have indicated its support of the Unambiguous SACK feature within the Supported Extensions Parameter as defined in [RFC5061]. If Unambiguous SACK has been negotiated on an association, Unambiguous SACKs MUST be returned whenever a SCTP receiver would return SACK information. If Unambiguous SACK has not been negotiated on an association, the RTX-ID field in the chunk header of incoming data chunks MUST be ignored and [RFC4960] SACK format and return policies MUST be adhered to.

Authors' Addresses

Karen E. E. Nielsen
Ericsson
Kistavaegen 25
Stockholm 164 80
Sweden

Email: karen.nielsen@tieto.com

Rafaelle De Santis
Ericsson
xx
xx xx
Italy

Email: rafaele.de.santis@ericsson.com

Anna Brunstrom
Karlstad University
Universitetsgatan 2
Karlstad 651 88
Sweden

Email: anna.brunstrom@kau.se

Michael Tuexen
Muenster Univ. of Appl. Science
Stegerwaldstrasse 39
Steinfurt 48565
Germany

Email: tuexen@fh-muenster.de

Randall Stewart
Netflix, Inc.
xx
Chapin 29036 SC
United States

Email: randall@lakerest.net

DOTS
Internet-Draft
Intended status: Standards Track
Expires: February 9, 2017

T. Reddy
D. Wing
P. Patil
M. Geller
Cisco
M. Boucadair
Orange

August 8, 2016

Co-operative DDoS Mitigation
draft-reddy-dots-transport-06

Abstract

This document specifies a mechanism that a DOTS client can use to signal that a network is under a Distributed Denial-of-Service (DDoS) attack to an upstream DOTS server so that appropriate mitigation actions are undertaken (including, blackhole, drop, rate-limit, or add to watch list) on the suspect traffic. The document specifies both DOTS signal and data channels. Happy Eyeballs considerations for the DOTS signal channel are also elaborated.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 9, 2017.

Table of Contents

1. Introduction

A distributed denial-of-service (DDoS) attack is an attempt to make machines or network resources unavailable to their intended users. In most cases, sufficient scale can be achieved by compromising enough end-hosts and using those infected hosts to perpetrate and amplify the attack. The victim in this attack can be an application server, a client, a router, a firewall, or an entire network, etc.

In a lot of cases, it may not be possible for an enterprise to determine the cause for an attack, but instead just realize that

certain resources seem to be under attack. The document proposes that, in such cases, the DOTS client just inform the DOTS server that the enterprise is under a potential attack and that the Mitigator monitor traffic to the enterprise to mitigate any possible attack. This document also describes a means for an enterprise, which act as DOTS clients, to dynamically inform its DOTS server of the IP addresses or prefixes that are causing DDoS. A Mitigator can use this information to discard flows from such IP addresses reaching the customer network.

The proposed mechanism can also be used between applications from various vendors that are deployed within the same network, some of them are responsible for monitoring and detecting attacks while others are responsible for enforcing policies on appropriate network elements. This cooperations contributes to a ensure a highly automated network that is also robust, reliable and secure. The advantage of this mechanism is that the DOTS server can provide protection to the DOTS client from bandwidth-saturating DDoS traffic.

How a Mitigator determines which network elements should be modified to install appropriate filtering rules is out of scope. A variety of mechanisms and protocols (including NETCONF [RFC6241]) may be considered to exchange information through a communication interface between the server and these underlying elements; the selection of appropriate mechanisms and protocols to be invoked for that interfaces is deployment-specific.

Terminology and protocol requirements for co-operative DDoS mitigation are obtained from DOTS requirements [I-D.ietf-dots-requirements]. This document satisfies all the use cases discussed in [I-D.ietf-dots-use-cases] except the Third-party DOTS notifications use case in Section 3.2.3 of [I-D.ietf-dots-use-cases] which is an optional feature and not a core use case. Third-party DOTS notifications are not part of the DOTS requirements document and the DOTS architecture [I-D.ietf-dots-architecture] does not assess whether that use case may have an impact on the architecture itself and/or trust model.

2. Notational Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

(D)TLS: For brevity this term is used for statements that apply to both Transport Layer Security [RFC5246] and Datagram Transport Layer Security [RFC6347]. Specific terms will be used for any statement that applies to either protocol alone.

3. Solution Overview

Network applications have finite resources like CPU cycles, number of processes or threads they can create and use, maximum number of simultaneous connections it can handle, limited resources of the control plane, etc. When processing network traffic, such an application uses these resources to offer its intended task in the most efficient fashion. However, an attacker may be able to prevent the application from performing its intended task by causing the application to exhaust the finite supply of a specific resource.

TCP DDoS SYN-flood, for example, is a memory-exhaustion attack on the victim and ACK-flood is a CPU exhaustion attack on the victim ([RFC4987]). Attacks on the link are carried out by sending enough traffic such that the link becomes excessively congested, and legitimate traffic suffers high packet loss. Stateful firewalls can also be attacked by sending traffic that causes the firewall to hold excessive state and the firewall runs out of memory, and can no longer instantiate the state required to pass legitimate flows. Other possible DDoS attacks are discussed in [RFC4732].

In each of the cases described above, the possible arrangements between the DOTS client and DOTS server to mitigate the attack are discussed in [I-D.ietf-dots-use-cases]. An example of network diagram showing a deployment of these elements is shown in Figure 1. Architectural relationship between DOTS agents is explained in [I-D.ietf-dots-architecture]. In this example, the DOTS server is operating on the access network.

```
  Network
  Resource         CPE router            Access network     __________
 +-----------+    +--------------+       +-------------+    /          \
 |           |____|              |_______|             |___ | Internet |
 |DOTS client|    | DOTS gateway |       | DOTS server |    |          |
 |           |    |              |       |             |    |          |
 +-----------+    +--------------+       +-------------+    \__________/
```

Figure 1

The DOTS server can also be running on the Internet, as depicted in Figure 2.

```
   Network                                                DDoS mitigation
   Resource        CPE router                 __________      service
  +-----------+    +-------------+       /          \    +-------------+
  |           |____|             |_______|          |___ |             |
  |DOTS client|    |DOTS gateway |       | Internet |    | DOTS server |
  |           |    |          |  |       |          |    |             |
  +-----------+    +-------------+       \__________/    +-------------+
```

Figure 2

In typical deployments, the DOTS client belongs to a different administrative domain than the DOTS server. For example, the DOTS client is a web server serving content owned and operated by an domain, while the DOTS server is owned and operated by a different domain providing DDoS mitigation services. That domain providing DDoS mitigation service might, or might not, also provide Internet access service to the website operator.

The DOTS server may (not) be co-located with the DOTS mitigator. In typical deployments, the DOTS server belongs to the same administrative domain as the mitigator.

The DOTS client can communicate directly with the DOTS server or indirectly with the DOTS server via a DOTS gateway.

4. Happy Eyeballs for DOTS Signal Channel

DOTS signaling can happen with DTLS [RFC6347] over UDP and TLS [RFC5246] over TCP. A DOTS client can use DNS to determine the IP address(es) of a DOTS server or a DOTS client may be provided with the list of DOTS server IP addresses. The DOTS client MUST know a DOTS server’s domain name; hard-coding the domain name of the DOTS server into software is NOT RECOMMENDED in case the domain name is not valid or needs to change for legal or other reasons. The DOTS client performs A and/or AAAA record lookup of the domain name and the result will be a list of IP addresses, each of which can be used to contact the DOTS server using UDP and TCP.

If an IPv4 path to reach a DOTS server is found, but the DOTS server’s IPv6 path is not working, a dual-stack DOTS client can experience a significant connection delay compared to an IPv4-only DOTS client. The other problem is that if a middlebox between the DOTS client and DOTS server is configured to block UDP, the DOTS client will fail to establish a DTLS session with the DOTS server and will, then, have to fall back to TLS over TCP incurring significant connection delays. [I-D.ietf-dots-requirements] discusses that DOTS client and server will have to support both connectionless and connection-oriented protocols.

To overcome these connection setup problems, the DOTS client can try connecting to the DOTS server using both IPv6 and IPv4, and try both DTLS over UDP and TLS over TCP in a fashion similar to the Happy Eyeballs mechanism [RFC6555]. These connection attempts are performed by the DOTS client when its initializes, and the client uses that information for its subsequent alert to the DOTS server. In order of preference (most preferred first), it is UDP over IPv6, UDP over IPv4, TCP over IPv6, and finally TCP over IPv4, which adheres to address preference order [RFC6724] and the DOTS preference that UDP be used over TCP (to avoid TCP's head of line blocking).

```
DOTS client                                                DOTS server
   |                                                           |
   |--DTLS ClientHello, IPv6 ---->X                            |
   |--TCP SYN, IPv6-------------->X                            |
   |--DTLS ClientHello, IPv4 ---->X                            |
   |--TCP SYN, IPv4------------------------------------------>|
   |--DTLS ClientHello, IPv6 ---->X                            |
   |--TCP SYN, IPv6-------------->X                            |
   |<-TCP SYNACK----------------------------------------------|
   |--DTLS ClientHello, IPv4 ---->X                            |
   |--TCP ACK-------------------------------------------------->|
   |<------------Establish TLS Session---------------------->|
   |----------------DOTS signal----------------------------->|
   |                                                           |
```

Figure 3: Happy Eyeballs

In reference to Figure 3, the DOTS client sends two TCP SYNs and two DTLS ClientHello messages at the same time over IPv6 and IPv4. In this example, it is assumed that the IPv6 path is broken and UDP is dropped by a middle box but has little impact to the DOTS client because there is no long delay before using IPv4 and TCP. The IPv6 path and UDP over IPv6 and IPv4 is retried until the DOTS client gives up.

5. DOTS Signal Channel

5.1. Overview

Constrained Application Protocol (CoAP) [RFC7252] is used for DOTS signal channel. COAP was designed according to the REST architecture, and thus exhibits functionality similar to that of HTTP, it is quite straightforward to map from CoAP to HTTP and from HTTP to CoAP. CoAP has been defined to make use of both DTLS over UDP and TLS over TCP. The advantages of COAP are: (1) Like HTTP, CoAP is based on the successful REST model, (2) CoAP is designed to

use minimal resources, (3) CoAP integrates with JSON, CBOR or any other data format, (4) asynchronous message exchanges, etc.

```
+--------------+
|    DOTS      |
+--------------+
|     CoAP     |
+--------------+
| TLS |  DTLS  |
+--------------+
| TCP |   UDP  |
+--------------+
|    IP        |
+--------------+
```

Figure 4: Abstract Layering of DOTS signal channel over CoAP over (D)TLS

JSON [RFC7159] payloads are used to convey signal channel specific payload messages that convey request parameters and response information such as errors.

TBD: Do we want to use CBOR [RFC7049] instead of JSON?

5.2. Mitigation Service Requests

The following APIs define the means to convey a DOTS signal from a DOTS client to a DOTS server:

POST requests: are used to convey the DOTS signal from a DOTS client to a DOTS server over the signal channel, possibly traversing a DOTS gateway, indicating the DOTS client’s need for mitigation, as well as the scope of any requested mitigation (Section 5.2.1). DOTS gateway act as a CoAP-to-CoAP Proxy (explained in [RFC7252]).

DELETE requests: are used by the DOTS client to withdraw the request for mitigation from the DOTS server (Section 5.2.2).

GET requests: are used by the DOTS client to retrieve the DOTS signal(s) it had conveyed to the DOTS server (Section 5.2.3).

PUT requests: are used by the DOTS client to convey mitigation efficacy updates to the DOTS server (Section 5.2.4).

Reliability is provided to the POST, DELETE, GET, and PUT requests by marking them as Confirmable (CON) messages. As explained in Section 2.1 of [RFC7252], a Confirmable message is retransmitted using a default timeout and exponential back-off between

retransmissions, until the DOTS server sends an Acknowledgement message (ACK) with the same Message ID conveyed from the DOTS client. Message transmission parameters are defined in Section 4.8 of [RFC7252]. Reliablity is provided to the responses by marking them as Confirmable (CON) messages. The DOTS server can either piggback the response in the acknowledgement message or if the DOTS server is not able to respond immediately to a request carried in a Confirmable message, it simply responds with an Empty Acknowledgement message so that the DOTS client can stop retransmitting the request. Empty Acknowledgement message is explained in Section 2.2 of [RFC7252]. When the response is ready, the server sends it in a new Confirmable message which then in turn needs to be acknowledged by the DOTS client (see Sections 5.2.1 and Sections 5.2.2 in [RFC7252]).

Implementation Note: A DOTS client that receives a response in a CON message may want to clean up the message state right after sending the ACK. If that ACK is lost and the DOTS server retransmits the CON, the DOTS client may no longer have any state to which to correlate this response, making the retransmission an unexpected message; the DOTS client will send a Reset message so it does not receive any more retransmissions. This behavior is normal and not an indication of an error (see Section 5.3.2 in [RFC7252] for more details).

5.2.1. Convey DOTS Signals

When suffering an attack and desiring DoS/DDoS mitigation, a DOTS signal is sent by the DOTS client to the DOTS server. A POST request is used to convey a DOTS signal to the DOTS server (Figure 5). The DOTS server can enable mitigation on behalf of the DOTS client by communicating the DOTS client’s request to the mitigator and relaying any mitigator feedback to the requesting DOTS client.

```
Header: POST (Code=0.02)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "DOTS-signal"
Uri-Path: "version"
Content-Type: "application/json"
{
   "policy-id": "integer",
   "target-ip": "string",
   "target-port": "string",
   "target-protocol": "string",
   "lifetime": "number"
 }
```

Figure 5: POST to convey DOTS signals

The header fields are described below.

policy-id: Identifier of the policy represented using a integer. This identifier MUST be unique for each policy bound to the DOTS client, i.e. ,the policy-id needs to be unique relative to the active policies with the DOTS server. This identifier must be generated by the DOTS client. This document does not make any assumption about how this identifier is generated. This is a mandatory attribute.

target-ip: A list of IP addresses or prefixes under attack. IP addresses and prefixes are separated by commas. Prefixes are represented using CIDR notation [RFC4632]. This is an optional attribute.

target-port: A list of ports under attack. Ports are seperated by commas and port number range (using "-"). For TCP, UDP, SCTP, or DCCP: the range of ports (e.g., 1024-65535). This is an optional attribute.

target-protocol: A list of protocols under attack. Valid protocol values include tcp, udp, sctp, and dccp. Protocol values are seperated by commas. This is an optional attribute.

lifetime: Lifetime of the mitigation request policy in seconds. Upon the expiry of this lifetime, and if the request is not refreshed, the mitigation request is removed. The request can be refreshed by sending the same request again. The default lifetime of the policy is 60 minutes -- this value was chosen to be long enough so that refreshing is not typically a burden on the DOTS client, while expiring the policy where the client has

unexpectedly quit in a timely manner. A lifetime of zero indicates indefinite lifetime for the mitigation request. The server MUST always indicate the actual lifetime in the response. This is an optional attribute in the request.

The relative order of two rules is determined by comparing their respective policy identifiers. The rule with lower numeric policy identifier value has higher precedence (and thus will match before) than the rule with higher numeric policy identifier value.

To avoid DOTS signal message fragmentation and the consequently decreased probability of message delivery, DOTS agents MUST ensure that the DTLS record MUST fit within a single datagram. If the Path MTU is not known to the DOTS server, an IP MTU of 1280 bytes SHOULD be assumed. The length of the URL MUST NOT exceed 256 bytes. If UDP is used to convey the DOTS signal and the request size exceeds the Path MTU then the DOTS client MUST split the DOTS signal into separate messages, for example the list of addresses in the 'target-ip' field could be split into multiple lists and each list conveyed in a new POST request.

Implementation Note: DOTS choice of message size parameters works well with IPv6 and with most of today's IPv4 paths. However, with IPv4, it is harder to absolutely ensure that there is no IP fragmentation. If IPv4 support on unusual networks is a consideration and path MTU is unknown, implementations may want to limit themselves to more conservative IPv4 datagram sizes such as 576 bytes, as per [RFC0791] IP packets up to 576 bytes should never need to be fragmented, thus sending a maximum of 500 bytes of DOTS signal over a UDP datagram will generally avoid IP fragmentation.

Figure 6 shows a POST request to signal that ports 80, 8080, and 443 on the servers 2002:db8:6401::1 and 2002:db8:6401::2 are being attacked.

```
Header: POST (Code=0.02)
Uri-Host: "www.example.com"
Uri-Path: ".well-known"
Uri-Path: "v1"
Uri-Path: "DOTS-signal"
Content-Type: "application/json"
{
  "policy-id":123321333242,
  "target-ip":[
      "2002:db8:6401::1",
      "2002:db8:6401::2"
  ],
  "target-port":[
      "80",
      "8080",
      "443"
  ],
  "target-protocol":"tcp"
}
```

Figure 6: POST for DOTS signal

The DOTS server indicates the result of processing the POST request using CoAP response codes. CoAP 2xx codes are success, CoAP 4xx codes are some sort of invalid request and 5xx codes are returned if the DOTS server has erred or is incapable of performing the mitigation. Response code 2.01 (Created) will be returned in the response if the DOTS server has accepted the mitigation request and will try to mitigate the attack. If the request is missing one or more mandatory attributes then 4.00 (Bad Request) will be returned in the response or if the request contains invalid or unknown parameters then 4.02 (Invalid query) will be returned in the response. The CoAP response will include the JSON body received in the request.

5.2.2. Withdraw a DOTS Signal

A DELETE request is used to withdraw a DOTS signal from a DOTS server (Figure 7).

```
Header: DELETE (Code=0.04)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-signal"
Content-Type: "application/json"
{
   "policy-id": "number"
}
```

Figure 7: Withdraw DOTS signal

If the DOTS server does not find the policy number conveyed in the DELETE request in its policy state data, then it responds with a 4.04 (Not Found) error response code. The DOTS server successfully acknowledges a DOTS client’s request to withdraw the DOTS signal using 2.02 (Deleted) response code, and ceases mitigation activity as quickly as possible.

5.2.3. Retrieving a DOTS Signal

A GET request is used to retrieve information and status of a DOTS signal from a DOTS server (Figure 8). If the DOTS server does not find the policy number conveyed in the GET request in its policy state data, then it responds with a 4.04 (Not Found) error response code.

1) To retrieve all DOTS signals signaled by the DOTS client.

```
Header: GET (Code=0.01)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-signal"
Uri-Path: "list"
Observe : 0
```

2) To retrieve a specific DOTS signal signaled by the DOTS client. The policy information in the response will be formatted in the same order it was processed at the DOTS server.

```
Header: GET (Code=0.01)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-signal"
Uri-Path: "policy-id value"
Observe : 0
```

Figure 8: GET to retrieve the rules

Figure 9 shows the response of all the active policies on the DOTS server.

```
{
  "policy-data":[
    {
      "policy-id":123321333242,
      "target-protocol":"tcp",
      "lifetime":3600,
      "status":"mitigation in progress"
    },
    {
      "policy-id":123321333244,
      "target-protocol":"udp",
      "lifetime":1800,
      "status":"mitigation complete"
    },
    {
      "policy-id":123321333245,
      "target-protocol":"tcp",
      "lifetime":1800,
      "status":"attack stopped"
    }
  ]
}
```


Figure 9: Response body

The various possible values of status field are explained below:

mitigation in progress: Attack mitigation is in progress (e.g., changing the network path to re-route the inbound traffic to DOTS mitigator).

mitigation complete: Attack is successfully mitigated (e.g., attack traffic is dropped).

attack stopped: Attack has stopped and the DOTS client can withdraw the mitigation request.

The observe option defined in [RFC7641] extends the CoAP core protocol with a mechanism for a CoAP client to "observe" a resource on a CoAP server: the client retrieves a representation of the resource and requests this representation be updated by the server as long as the client is interested in the resource. A DOTS client conveys the observe option set to 0 in the GET request to receive unsolicited notifications of attack mitigation status from the DOTS server. Unidirectional notifications within the bidirectional signal channel allows unsolicited message delivery, enabling asynchronous notifications between the agents. A DOTS client that is no longer interested in receiving notifications from the DOTS server can simply

"forget" the observation. When the DOTS server then sends the next notification, the DOTS client will not recognize the token in the message and thus will return a Reset message. This causes the DOTS server to remove the associated entry.

```
                     DOTS Client            DOTS Server
                        |                          |
                        |  GET /<policy-id number>  |
                        |  Token: 0x4a              |   Registration
                        |  Observe: 0               |
                        +-------------------------->|
                        |                          |
                        |  2.05 Content             |
                        |  Token: 0x4a              |   Notification of
                        |  Observe: 12              |   the current state
                        |  status: "mitigation      |
                        |          in progress"     |
                        |<--------------------------+
                        |  2.05 Content             |
                        |  Token: 0x4a              |   Notification upon
                        |  Observe: 44              |    a state change
                        |  status: "mitigation      |
                        |          complete"        |
                        |<--------------------------+
                        |  2.05 Content             |
                        |  Token: 0x4a              |   Notification upon
                        |  Observe: 60              |   a state change
                        |  status: "attack stopped" |
                        |<--------------------------+
                        |                          |
```

Figure 10: Notifications of attack mitigation status

5.2.3.1. Mitigation Status

A DOTS client retrieves the information about a DOTS signal at frequent intervals to determine the status of an attack. If the DOTS server has been able to mitigate the attack and the attack has stopped, the DOTS server indicates as such in the status, and the DOTS client recalls the mitigation request.

A DOTS client should react to the status of the attack from the DOTS server and not the fact that it has recognized, using its own means, that the attack has been mitigated. This ensures that the DOTS client does not recall a mitigation request in a premature fashion because it is possible that the DOTS client does not sense the DDOS attack on its resources but the DOTS server could be actively mitigating the attack and the attack is not completely averted.

5.2.4. Efficacy Update from DOTS Client

While DDoS mitigation is active, a DOTS client MAY frequently transmit DOTS mitigation efficacy updates to the relevant DOTS server. An PUT request (Figure 11) is used to convey the mitigation efficacy update to the DOTS server. The PUT request MUST include all the header fields used in the POST request to convey the DOTS signal (Section 5.2.1). If the DOTS server does not find the policy number conveyed in the PUT request in its policy state data, it responds with a 4.04 (Not Found) error response code.

```
Header: PUT (Code=0.03)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-signal"
Uri-Path: "policy-id value"
Content-Type: "application/json"
{
  "target-ip": "string",
  "target-port": "string",
  "target-protocol": "string",
  "lifetime": "number",
  "attack-status": "string"
 }
```

Figure 11: Efficacy Update

The ’attack-status’ field is a mandatory attribute. The various possible values contained in the ’attack-status’ field are explained below:

in-progress: DOTS client determines that it is still under attack.

terminated: Attack is successfully mitigated (e.g., attack traffic is dropped).

6. DOTS Data Channel

Note: Based on discussions at IETF-96 DOTS implementers meeting, in later revision this section becomes its own stand-alone specification and will include https://tools.ietf.org/html/draft-nishizuka-dots-inter-domain-mechanism-01.

The DOTS data channel is intended to be used for bulk data exchanges between DOTS agents. Unlike the signal channel, which must operate nominally even when confronted with despite signal degradation due to packet loss, the data channel is not expected to be constructed to

deal with attack conditions. As the primary function of the data channel is data exchange, a reliable transport is required in order for DOTS agents to detect data delivery success or failure. CoAP over TLS over TCP is used for DOTS data channel.

```
+--------------+
|    DOTS      |
+--------------+
|     CoAP     |
+--------------+
|     TLS      |
+--------------+
|     TCP      |
+--------------+
|     IP       |
+--------------+
```

Figure 12: Abstract Layering of DOTS data channel over CoAP over TLS

JSON payloads is used to convey both filtering rules as well as data channel specific payload messages that convey request parameters and response information such as errors. All data channel URIs defined in this document, and in subsequent documents, MUST NOT have a URI containing "/DOTS-signal".

One of the possible arrangements for DOTS client to signal filtering rules to a DOTS server via the DOTS gateway is discussed below:

The DOTS data channel conveys the filtering rules to the DOTS gateway. The DOTS gateway validates if the DOTS client is authorized to signal the filtering rules and if the client is authorized propagates the rules to the DOTS server. Likewise, the DOTS server validates if the DOTS gateway is authorized to signal the filtering rules. To create or purge filters, the DOTS client sends CoAP requests to the DOTS gateway. The DOTS gateway acts as a proxy, validates the rules and proxies the requests containing the filtering rules to a DOTS server. When the DOTS gateway receives the associated CoAP response from the DOTS server, it propagates the response back to the DOTS client.

6.1. Filtering Rules

The following APIs define means for a DOTS client to configure filtering rules on a DOTS server.

6.1.1. Install Filtering Rules

An POST request is used to push filtering rules to a DOTS server (Figure 13).

```
Header: POST (Code=0.02)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-data-channel"
Content-Type: "application/json"
{
   "policy-id": "integer",
   "traffic-protocol": "string",
   "source-protocol-port": "string",
   "destination-protocol-port": "string",
   "destination-ip": "string",
   "source-ip": "string",
   "lifetime": "number",
   "traffic-rate" : "number"
 }
```

Figure 13: POST to install filtering rules

The header fields are described below:

policy-id: Identifier of the policy represented using a integer. This identifier MUST be unique for each policy bound to the DOTS client, i.e., the policy-id needs to be unique relative to the active policies with the DOTS server. This identifier must be generated by the client. This document does not make any assumption about how this identifier is generated. This is an mandatory attribute.

traffic-protocol: Valid protocol values include tcp, udp, sctp, and dccp. Protocol values are seperated by commas (e.g. "tcp, udp"). This is an mandatory attribute.

source-protocol-port: The source port number. Ports are seperated by commas and port number range (using "-"). For TCP, UDP, SCTP, or DCCP: the source range of ports (e.g., 1024-65535). This is an optional attribute.

destination-protocol-port: The destination port number. Ports are seperated by commas and port number range (using "-"). For TCP, UDP, SCTP, or DCCP: the destination range of ports (e.g., 443-443). This information is useful to avoid disturbing a group

of customers when address sharing is in use [RFC6269]. This is an optional attribute.

destination-ip: The destination IP address or prefix. IP addresses and prefixes are separated by commas. Prefixes are represented using CIDR notation. This is an optional attribute.

source-ip: The source IP addresses or prefix. IP addresses and prefixes are separated by commas. Prefixes are represented using CIDR notation. This is an optional attribute.

lifetime: Lifetime of the rule in seconds. Upon the expiry of this lifetime, and if the request is not refreshed, this particular rule is removed. The rule can be refreshed by sending the same message again. The default lifetime of the rule is 60 minutes -- this value was chosen to be long enough so that refreshing is not typically a burden on the DOTS client, while expiring the rule where the client has unexpectedly quit in a timely manner. A lifetime of zero indicates indefinite lifetime for the rule. The server MUST always indicate the actual lifetime in the response. This is an optional attribute in the request.

traffic-rate: This is the allowed traffic rate in bytes per second indicated in IEEE floating point [IEEE.754.1985] format. The value 0 indicates all traffic for the particular flow to be discarded. This is a mandatory attribute.

The relative order of two rules is determined by comparing their respective policy identifiers. The rule with lower numeric policy identifier value has higher precedence (and thus will match before) than the rule with higher numeric policy identifier value.

Figure 14 shows a POST request to block traffic from attacker IPv6 prefix 2001:db8:abcd:3f01::/64 to network resource using IPv6 address 2002:db8:6401::1 to operate a server on TCP port 443.

```
Header: POST (Code=0.02)
Uri-Host: "www.example.com"
Uri-Path: ".well-known"
Uri-Path: "v1"
Uri-Path: "DOTS-data-channel"
Content-Type: "application/json"
 {
   "policy-id": 123321333242,
   "traffic-protocol": "tcp",
   "source-protocol-port": "0-65535",
   "destination-protocol-port": "443",
   "destination-ip": "2001:db8:abcd:3f01::/64",
   "source-ip": "2002:db8:6401::1",
   "lifetime": 1800,
   "traffic-rate": 0
 }
```

Figure 14: POST to Install Black-list Rules

6.1.2. Remove Filtering Rules

A DELETE request is used to delete filtering rules from a DOTS server (Figure 15).

```
Header: DELETE (Code=0.04)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-data-channel"
Content-Type: "application/json"
 {
   "policy-id": "number"
 }
```

Figure 15: DELETE to remove the rules

6.1.3. Retrieving Installed Filtering Rules

The DOTS client periodically queries the DOTS server to check the counters for installed filtering rules. A GET request is used to retrieve filtering rules from a DOTS server.

Figure 16 shows an example to retrieve all the filtering rules programmed by the DOTS client while Figure 17 shows an example to retrieve specific filtering rules programmed by the DOTS client.

```
Header: GET (Code=0.01)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-data-channel"
Uri-Path: "list"
```

Figure 16: GET to retrieve the rules (1)

```
Header: GET (Code=0.01)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-data-channel"
Uri-Path: "policy-id value"
```

Figure 17: GET to retrieve the rules (2)

Figure 18 shows response for all active policies on the DOTS server.


```
{
  "policy-data":[
    {
      "policy-id":123321333242,
      "traffic-protocol": "tcp",
      "source-protocol-port": "0-65535",
      "destination-protocol-port": "443",
      "destination-ip": "2001:db8:abcd:3f01::/64",
      "source-ip": "2002:db8:6401::1",
      "lifetime": 1800,
      "traffic-rate": 0,
      "match-count": 689324,
    },
    {
      "policy-id":123321333242,
      "traffic-protocol": "udp",
      "source-protocol-port": "0-65535",
      "destination-protocol-port": "53",
      "destination-ip": "2001:db8:abcd:3f01::/64",
      "source-ip": "2002:db8:6401::2",
      "lifetime": 1800,
      "traffic-rate": 0,
      "match-count": 6666,
    }
  ]
}
```


Figure 18: Response body

7. (D)TLS Protocol Profile and Performance considerations

This section defines the (D)TLS protocol profile of DOTS signal channel over (D)TLS and DOTS data channel over TLS.

There are known attacks on (D)TLS, such as machine-in-the-middle and protocol downgrade. These are general attacks on (D)TLS and not specific to DOTS over (D)TLS; please refer to the (D)TLS RFCs for discussion of these security issues. DOTS agents MUST adhere to the (D)TLS implementation recommendations and security considerations of [RFC7525] except with respect to (D)TLS version. Since encryption of DOTS using (D)TLS is virtually a green-field deployment DOTS agents MUST implement only (D)TLS 1.2 or later.

Implementations compliant with this profile MUST implement all of the following items:

- o DOTS client can use (D)TLS session resumption without server-side state [RFC5077] to resume session and convey the DOTS signal.

- o While the communication to the DOTS server is quiescent, the DOTS client MAY probe the server to ensure it has maintained cryptographic state. Such probes can also keep alive firewall or NAT bindings. This probing reduces the frequency of needing a new handshake when a DOTS signal needs to be conveyed to the DOTS server.

  - * A (D)TLS heartbeat [RFC6520] verifies the DOTS server still has DTLS state by returning a DTLS message. If the server has lost state, it returns a DTLS Alert. Upon receipt of an unauthenticated DTLS Alert, the DTLS client validates the Alert is within the replay window (Section 4.1.2.6 of [RFC6347]). It is difficult for the DTLS client to validate the DTLS Alert was generated by the DTLS server in response to a request or was generated by an on- or off-path attacker. Thus, upon receipt of an in-window DTLS Alert, the client SHOULD continue re-transmitting the DTLS packet (in the event the Alert was spoofed), and at the same time it SHOULD initiate DTLS session resumption.

  - * TLS runs over TCP, so a simple probe is a 0-length TCP packet (a "window probe"). This verifies the TCP connection is still working, which is also sufficient to prove the server has retained TLS state, because if the server loses TLS state it abandons the TCP connection. If the server has lost state, a TCP RST is returned immediately.

* Raw public keys [RFC7250] which reduce the size of the ServerHello, and can be used by servers that cannot obtain certificates (e.g., DOTS gateways on private networks).

Implementations compliant with this profile SHOULD implement all of the following items to reduce the delay required to deliver a DOTS signal:

- o TLS False Start [I-D.ietf-tls-falsestart] which reduces round-trips by allowing the TLS second flight of messages (ChangeCipherSpec) to also contain the DOTS signal.

- o Cached Information Extension [I-D.ietf-tls-cached-info] which avoids transmitting the server’s certificate and certificate chain if the client has cached that information from a previous TLS handshake.

- o TCP Fast Open [RFC7413] can reduce the number of round-trips to convey DOTS signal.

8. Mutual Authentication of DOTS Agents & Authorization of DOTS Clients

(D)TLS based on client certificate can be used for mutual authentication between DOTS agents. If a DOTS gateway is involved, DOTS clients and DOTS gateway MUST perform mutual authentication; only authorized DOTS clients are allowed to send DOTS signals to a DOTS gateway. DOTS gateway and DOTS server MUST perform mutual authentication; DOTS server only allows DOTS signals from authorized DOTS gateway, creating a two-link chain of transitive authentication between the DOTS client and the DOTS server.

```
 +-------------------------------------------------+
 |        example.com domain          +---------+  |
 |                                    | AAA     |  |
 |   +---------------+                | Server  |  |
 |   | Application   |                +------+--+  |
 |   | server        +                       ^
 |   | (DOTS client) |<-----------------+    |     |
 |   +---------------+                  +    |     |                example.net
domain
 |                                      V    V     |
 |                              +-------------+   |              +-------------
--+
 |  +--------------+            |             |   |              |
  |
 |  |   Guest      +<-----x----->+             +<----------------->+    DOTS
  |
 |  | (DOTS client)|             |   DOTS      |   |              |    Server
  |
 |  +--------------+            |   Gateway   |   |              |
  |
 |                              +----+--------+   |              +-------------
--+
 |                                   ^            |
 |                                   |            |
 |   +----------------+              |            |
 |   | DDOS detector  |              |            |
 |   | (DOTS client)  +<--------------+           |
 |   +----------------+                           |
 |                                                |
 +-------------------------------------------------+
```

Figure 19: Example of Authentication and Authorization of DOTS Agents

In the example depicted in Figure 19, the DOTS gateway and DOTS clients within the 'example.com' domain mutually authenticate with each other. After the DOTS gateway validates the identity of a DOTS client, it communicates with the AAA server in the 'example.com' domain to determine if the DOTS client is authorized to request DDOS mitigation. If the DOTS client is not authorized, a 4.01 (Unauthorized) is returned in the response to the DOTS client. In this example, the DOTS gateway only allows the application server and DDOS detector to request DDOS mitigation, but does not permit the user of type 'guest' to request DDOS mitigation.

Also, DOTS gateway and DOTS server MUST perform mutual authentication using certificates. A DOTS server will only allow a DOTS gateway with a certificate for a particular domain to request mitigation for that domain. In reference to Figure 19, the DOTS server only allows the DOTS gateway to request mitigation for 'example.com' domain and not for other domains.

9. IANA Considerations

TODO

[TBD: DOTS WG will probably have to do something similar to https://tools.ietf.org/html/rfc7519#section-10, create JSON DOTS claim registry and register the JSON attributes defined in this specification].

10. Security Considerations

Authenticated encryption MUST be used for data confidentiality and message integrity. (D)TLS based on client certificate MUST be used for mutual authentication. The interaction between the DOTS agents requires Datagram Transport Layer Security (DTLS) and Transport Layer Security (TLS) with a ciphersuite offering confidentiality protection and the guidance given in [RFC7525] MUST be followed to avoid attacks on (D)TLS.

If TCP is used between DOTS agents, attacker may be able to inject RST packets, bogus application segments, etc., regardless of whether TLS authentication is used. Because the application data is TLS protected, this will not result in the application receiving bogus data, but it will constitute a DoS on the connection. This attack can be countered by using TCP-AO [RFC5925]. If TCP-AO is used, then any bogus packets injected by an attacker will be rejected by the TCP-AO integrity check and therefore will never reach the TLS layer.

Special care should be taken in order to ensure that the activation of the proposed mechanism won’t have an impact on the stability of the network (including connectivity and services delivered over that network).

Involved functional elements in the cooperation system must establish exchange instructions and notification over a secure and authenticated channel. Adequate filters can be enforced to avoid that nodes outside a trusted domain can inject request such as deleting filtering rules. Nevertheless, attacks can be initiated from within the trusted domain if an entity has been corrupted. Adequate means to monitor trusted nodes should also be enabled.

11. Contributors

Robert Moskowitz

12. Acknowledgements

Thanks to Christian Jacquenet, Roland Dobbins, Andrew Mortensen, Roman D. Danyliw, and Gilbert Clark for the discussion and comments.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <http://www.rfc-editor.org/info/rfc7250>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <http://www.rfc-editor.org/info/rfc7641>.

13.2. Informative References

[I-D.ietf-dots-architecture]
Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-00 (work in progress), July 2016.

[I-D.ietf-dots-requirements]
Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-02 (work in progress), July 2016.

[I-D.ietf-dots-use-cases]
Dobbins, R., Fouant, S., Migault, D., Moskowitz, R., Teague, N., and L. Xia, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-01 (work in progress), March 2016.

[I-D.ietf-tls-cached-info]
Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", draft-ietf-tls-cached-info-23 (work in progress), May 2016.

[I-D.ietf-tls-falsestart]
Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", draft-ietf-tls-falsestart-02 (work in progress), May 2016.

[IEEE.754.1985]
Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", August 1985.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <http://www.rfc-editor.org/info/rfc4632>.

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, DOI 10.17487/RFC4732, December 2006, <http://www.rfc-editor.org/info/rfc4732>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <http://www.rfc-editor.org/info/rfc4987>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <http://www.rfc-editor.org/info/rfc5077>.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <http://www.rfc-editor.org/info/rfc5575>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <http://www.rfc-editor.org/info/rfc6269>.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, DOI 10.17487/RFC6520, February 2012, <http://www.rfc-editor.org/info/rfc6520>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <http://www.rfc-editor.org/info/rfc7413>.

Authors' Addresses

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

Prashanth Patil
Cisco Systems, Inc.

Email: praspati@cisco.com

Mike Geller
Cisco Systems, Inc.
3250
Florida 33309
USA

Email: mgeller@cisco.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

TSVWG
Internet Draft
Intended status: Standards Track
Intended updates: 768
Expires: November 2017

J. Touch
USC/ISI

May 16, 2017

Transport Options for UDP
draft-touch-tsvwg-udp-options-09.txt

Status of this Memo

Copyright Notice

Abstract

Transport protocols are extended through the use of transport header options. This document experimentally extends UDP by indicating the location, syntax, and semantics for UDP transport layer options.

Table of Contents

1. Introduction

Transport protocols use options as a way to extend their capabilities. TCP [RFC793], SCTP [RFC4960], and DCCP [RFC4340] include space for these options but UDP [RFC768] currently does not. This document defines an experimental extension to UDP that provides space for transport options including their generic syntax and

semantics for their use in UDP’s stateless, unreliable message protocol.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lowercase uses of these words are not to be interpreted as carrying significance described in RFC 2119.

In this document, the characters ">>" preceding an indented line(s) indicates a statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the portions of this RFC covered by these key words.

3. Background

Many protocols include a default header and an area for header options. These options enable the protocol to be extended for use in particular environments or in ways unforeseen by the original designers. Examples include TCP’s Maximum Segment Size, Window Scale, Timestamp, and Authentication Options [RFC793][RFC5925][RFC7323].

These options are used both in stateful (connection-oriented, e.g., TCP [RFC793], SCTP [RFC4960], DCCP [RFC4340]) and stateless (connectionless, e.g., IPv4 [RFC791], IPv6 [RFC2460] protocols. In stateful protocols they can help extend the way in which state is managed. In stateless protocols their effect is often limited to individual packets, but they can have an aggregate effect on a sequence as well. One example of such uses is Substrate Protocol for User Datagrams (SPUD) [Tr15], and this document is intended to provide an out-of-band option area as an alternative to the in-band mechanism currently proposed [Hi15].

UDP is one of the most popular protocols that lacks space for options [RFC768]. The UDP header was intended to be a minimal addition to IP, providing only ports and a data checksum for protection. This document experimentally extends UDP to provide a trailer area for options located after the UDP data payload.

4. The UDP Option Area

The UDP transport header includes demultiplexing and service identification (port numbers), a checksum, and a field that indicates the UDP datagram length (including UDP header). The UDP Length length field is typically redundant with the size of the maximum space available as a transport protocol payload (see also discussion in Section 9).

For IPv4, IP Total Length field indicates the total IP datagram length (including IP header), and the size of the IP options is indicated in the IP header (in 4-byte words) as the "Internet Header Length" (IHL), as shown in Figure 1 [RFC791]. As a result, the typical (and largest valid) value for UDP Length is:

```
UDP_Length = IPv4_Total_Length - IPv4_IHL * 4
```

For IPv6, the IP Payload Length field indicates the datagram after the base IPv6 header, which includes the IPv6 extension headers and space available for the transport protocol, as shown in Figure 2 [RFC2460]. Note that the Next HDR field in IPv6 might not indicate UDP (i.e., 17), e.g., when intervening IP extension headers are present. For IPv6, the lengths of any additional IP extensions are indicated within each extension [RFC2460], so the typical (and largest valid) value for UDP Length is:

```
UDP_Length = IPv6_Payload_Length - sum(extension header lengths)
```

In both cases, the space available for the UDP transport protocol data unit is indicated by IP, either completely in the base header (for IPv4) or adding information in the extensions (for IPv6). In either case, this document will refer to this available space as the "IP transport payload".

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live | Proto=17 (UDP)|        Header Checksum        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
... zero or more IP Options (using space as indicated by IHL) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         UDP Source Port       |     UDP Destination Port      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          UDP Length           |         UDP Checksum          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1 IPv4 datagram with UDP transport payload

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |             Flow Label                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |   Next Hdr    |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
|                       Source Address (128 bits)               |
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
|                    Destination Address (128 bits)             |
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
... zero or more IP Extension headers (each indicating size)  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         UDP Source Port       |     UDP Destination Port      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          UDP Length           |         UDP Checksum          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2 IPv6 datagram with UDP transport payload

As a result of this redundancy, there is an opportunity to use the UDP Length field as a way to break up the IP transport payload into two areas - that intended as UDP user data and an additional "surplus area" (as shown in Figure 3).

```
                          IP transport payload
             <------------------------------------------------>
   +--------+---------+----------------------+------------------+
   | IP Hdr | UDP Hdr |     UDP user data    |   surplus area   |
   +--------+---------+----------------------+------------------+
             <------------------------------>
                        UDP Length
```

Figure 3 IP transport payload vs. UDP Length

In most cases, the IP transport payload and UDP Length point to the same location, indicating that there is no surplus area. It is important to note that this is not a requirement of UDP [RFC768] (discussed further in Section 9). UDP-Lite used the difference in these pointers to indicate the partial coverage of the UDP Checksum, such that the UDP user data, UDP header, and UDP pseudoheader (a subset of the IP header) are covered by the UDP checksum but additional user data in the surplus area is not covered [RFC3828]. This document uses the surplus area for UDP transport options.

The UDP option area is thus defined as the location between the end of the UDP payload and the end of the IP datagram as a trailing options area. This area can occur at any valid byte offset, i.e., it need not be 16-bit or 32-bit aligned. In effect, this document redefines the UDP "Length" field as a "trailer offset".

UDP options are defined using a TLV (type, length, and optional value) syntax similar to that of TCP [RFC793]. They are typically a minimum of two bytes in length as shown in Figure 4, excepting only the one byte options "No Operation" (NOP) and "End of Options List" (EOL) described below.

```
+--------+--------+
|  Kind  | Length |
+--------+--------+
```

Figure 4 UDP option default format

>> UDP options MAY occur at any UDP length offset.

>> The UDP length MUST be at least as large as the UDP header (8) and no larger than the IP transport payload. Values outside this range MUST be silently discarded as invalid and logged where rate-limiting permits.

Others have considered using values of the UDP Length that is larger than the IP transport payload as an additional type of signal. Using

a value smaller than the IP transport payload is expected to be backward compatible with existing UDP implementations, i.e., to deliver the UDP Length of user data to the application and silently ignore the additional surplus area data. Using a value larger than the IP transport payload would either be considered malformed (and be silently dropped) or could cause buffer overruns, and so is not considered silently and safely backward compatible. Its use is thus out of scope for the extension described in this document.

>> UDP options MUST be interpreted in the order in which they occur in the UDP option area.

5. UDP Options

The following UDP options are currently defined:

Kind	Length	Meaning
0*	-	End of Options List (EOL)
1*	-	No operation (NOP)
2*	2	Option checksum (OCS)
3	4	Alternate checksum (ACS)
4	4	Lite (LITE)
5	4	Maximum segment size (MSS)
6	10	Timestamps (TIME)
7	12	Fragmentation (FRAG)
8	(varies)	Authentication and Encryption (AE)
9-126	(varies)	UNASSIGNED (assignable by IANA)
127-253		RESERVED
254	N(>=4)	RFC 3692-style experiments (EXP)
255		RESERVED

These options are defined in the following subsections.

>> An endpoint supporting UDP options MUST support those marked with a "*" above: EOL, NOP, and OCS.

[QUESTION: Should we extend these, e.g., through #7?]

>> All other options (without a "*") MAY be implemented, and their use SHOULD be determined either out-of-band or negotiated.

>> Receivers MUST silently ignore unknown options. That includes options whose length does not indicate the specified value.

Receivers cannot treat unexpected option lengths as invalid, as this would unnecessarily limit future revision of options (e.g., defining a new ACS that is defined by having a different length).

>> Option lengths MUST NOT exceed the IP length of the packet. If this occurs, the packet MUST be treated as malformed and dropped, and the event MAY be logged for diagnostics (logging SHOULD be rate limited).

>> Required options MUST come before other options. Each required option MUST NOT occur more than once (if they are repeated in a received segment, all except the first MUST be silently ignored).

The requirement that required options come before others is intended to allow for endpoints to implement DOS protection, as discussed further in Section 12.

5.1. End of Options List (EOL)

The End of Options List (EOL) option indicates that there are no more options. It is used to indicate the end of the list of options without needing to pad the options to fill all available option space.

```
+--------+
| Kind=0 |
+--------+
```

Figure 5 UDP EOL option format

>> When the UDP options do not consume the entire option area, the last non-NOP option SHOULD be EOL (vs. filling the entire option area with NOP values).

>> All bytes after EOL MUST be ignored by UDP option processing. As a result, there can only ever be one EOL option (even if other bytes were zero, they are ignored).

5.2. No Operation (NOP)

The No Operation (NOP) option is a one byte placeholder, intended to be used as padding, e.g., to align multi-byte options along 16-bit or 32-bit boundaries.

```
+--------+
| Kind=1 |
+--------+
```

Figure 6 UDP NOP option format

>> If options longer than one byte are used, NOP options SHOULD be used at the beginning of the UDP options area to achieve alignment as would be more efficient for active (i.e., non-NOP) options.

>> Segments SHOULD NOT use more than three consecutive NOPs. NOPs are intended to assig with alignment, not other padding or fill.

[NOTE: Tom Herbert suggested we declare "more than 3 consecutive NOPs" a fatal error to reduce the potential of using NOPs as a DOS attack, but IMO there are other equivalent ways (e.g., using RESERVED or other UNASSIGNED values) and the "no more than 3" creates its own DOS vulnerability)

5.3. Option Checksum (OCS)

The Option Checksum (OCS) is an 8-bit ones-complement sum (Ones8) that covers all of the UDP options. OCS is 8-bits to allow the entire option to occupy a total of 16 bits.

OCS can be calculated by computing the 16-bit ones-complement sum and "folding over" the result (using carry wraparound). Note that OCS is direct, i.e., it is not negated or adjusted if zero (unlike the Internet checksum as used in IPv4, TCP, and UDP headers). OCS protects the option area from errors in a similar way that the UDP checksum protects the UDP user data.

```
+--------+--------+
| Kind=2 | Ones8  |
+--------+--------+
```

Figure 7 UDP OCS option format

>> When present, the option checksum SHOULD occur as early as possible, preferably preceded by only NOP options for alignment and the LITE option if present.

OCS covers the entire UDP option, including the Lite option as formatted before swapping for transmission (or, equivalently, after the swap after reception).

>> If the option checksum fails, all options MUST be ignored and any trailing surplus data (and Lite data, if used) silently discarded.

>> UDP data that is validated by a correct UDP checksum MUST be delivered to the application layer, even if the UDP option checksum fails, unless the endpoints have negotiated otherwise for this segment’s socket pair.

5.4. Alternate Checksum (ACS)

The Alternate Checksum (ACS) is a 16-bit CRC of the UDP payload only (excluding the IP pseudoheader, UDP header, and UDP options). It does not include the IP pseudoheader or UDP header, and so need not be updated by NATs when IP addresses or UDP ports are rewritten. Its purpose is to detect errors that the UDP checksum might not detect. CRC-CCITT (polynomial x^16 + x^12 + x^5 + x or polynomial 0x1021) has been chosen because of its ubiquity and use in other packet protocols, such as X.25, HDLC, and Bluetooth.

```
+--------+--------+--------+--------+
| Kind=3 | Len=4  |     CRC16sum    |
+--------+--------+--------+--------+
```

Figure 8 UDP ACS option format

5.5. Lite (LITE)

The Lite option (LITE) is intended to provide equivalent capability to the UDP Lite transport protocol [RFC3828]. UDP Lite allows the UDP checksum to cover only a prefix of the UDP data payload, to protect critical information (e.g., application headers) but allow potentially erroneous data to be passed to the user. This feature helps protect application headers but allows for application data errors. Some applications are impacted more by a lack of data than errors in data, e.g., voice and video.

>> When LITE is active, it MUST come first in the UDP options list.

LITE is intended to support the same API as for UDP Lite to allow applications to send and receive data that has a marker indicating the portion protected by the UDP checksum and the portion not protected by the UDP checksum.

LITE includes a 2-byte offset that indicates the length of the portion of the UDP data that is not covered by the UDP checksum.

```
+--------+--------+--------+--------+
| Kind=5 | Len=4  |      Offset     |
+--------+--------+--------+--------+
```

Figure 9 UDP LITE option format

At the sender, the option is formed using the following steps:

1. Create a LITE option, ordered as the first UDP option (Figure 10).

2. Calculate the location of the start of the options as an absolute offset from the start of the UDP header and place that length in the last two bytes of the LITE option.

3. Swap all four bytes of the LITE option with the first 4 bytes of the LITE data area (Figure 11).

```
+---------+--------------+--------------+------------------+
| UDP Hdr |  user data   |  LITE data   |LITE| other opts  |
+---------+--------------+--------------+------------------+
 <----------------------->
        UDP Length
```

Figure 10 LITE option formation - LITE goes first

```
+---------+--------------+--------------+------------------+
| UDP Hdr |  user data   |  LITE data   |LITE| other opts  |
+---------+--------------+--------------+------------------+
                          ^^^^           ^^^^
                            |              |
                            +--------------+
```

Figure 11 Before sending swap LITE option and front of LITE data

The resulting packet has the format shown in Figure 12. Note that the UDP length now points to the LITE option, and the LITE option points to the start of the option area.

```
+---------+--------------+----------------+------------------+
| UDP Hdr |  user data   |LITE| LITE data |Ldat| other opts  |
+---------+--------------+----------------+------------------+
 <---------------------->    |             ^
        UDP Length           +-------------+
```

Figure 12 Lite option as sent

A legacy endpoint receiving this packet will discard the LITE option and everything that follows, including the lite data and remainder of the UDP options. The UDP checksum will protect only the user data, not the LITE option or lite data.

Receiving endpoints capable of processing UDP options will do the following:

1. Process options as usual. This will start at the LITE option.

2. When the LITE option is encountered, record its location as the start of the LITE data area and swap the four bytes there with the four bytes at the location indicated inside the LITE option, which indicates the start of all of the options, including the LITE one (one past the end of the lite data area). This restores the format of the option as per Figure 10.

3. Continue processing the remainder of the options, which are now in the format shown in Figure 11.

The purpose of this swap is to support the equivalent of UDP Lite operation together with other UDP options without requiring the entire LITE data area to be moved after the UDP option area.

5.6. Maximum Segment Size (MSS)

The Maximum Segment Size (MSS, Kind = 3) is a 16-bit indicator of the largest UDP segment that can be received. As with the TCP MSS option [RFC793], the size indicated is the IP layer MTU decreased by the fixed IP and UDP headers only [RFC6691]. The space needed for IP and UDP options need to be adjusted by the sender when using the value indicated. The value transmitted is based on EMTU_R, the largest IP datagram that can be received (i.e., reassembled at the receiver) [RFC1122].

```
+--------+--------+--------+--------+
| Kind=5 | Len=4  |    MSS size     |
+--------+--------+--------+--------+
```

Figure 13 UDP MSS option format

The UDP MSS option MAY be used for path MTU discovery [RFC1191][RFC1981], but this may be difficult because of known issues with ICMP blocking [RFC2923] as well as UDP lacking automatic retransmission. It is more likely to be useful when coupled with IP source fragmentation to limit the largest reassembled UDP message, e.g., when EMTU_R is larger than the required minimums (576 for IPv4 [RFC791] and 1500 for IPv6 [RFC2460]).

5.7. Timestamps (TIME)

The UDP Timestamp option (TIME) exchanges two four-byte timestamp fields. It serves a similar purpose to TCP's TS option [RFC7323], enabling UDP to estimate the round trip time (RTT) between hosts. For UDP, this RTT can be useful for establishing UDP fragment reassembly timeouts or transport-layer rate-limiting [RFC8085].

```
+--------+--------+-----------------+-----------------+
| Kind=6 | Len=10 |     TS Value    |   TS Echo Reply |
+--------+--------+-----------------+-----------------+
  1 byte   1 byte       4 bytes            4 bytes
```

Figure 14 UDP TIME option format

TS Value (TSval) and TS Echo (TSecr) are used in a similar manner to the TCP TS option [RFC7323]. A host using the Timestamp option sets TS Value on all UDP segments issued. Received TSval values are provided to the application, which passes this value as TSecr on UDP messages sent in response to such a message.

>> UDP MAY use an RTT estimate based on nonzero Timestamp values as a hint for fragmentation reassembly, rate limiting, or other mechanisms that benefit from such an estimate.

>> UDP SHOULD make this RTT estimate available to the user application.

5.8. Fragmentation (FRAG)

The Fragmentation option (FRAG) supports UDP fragmentation and reassembly, which can be used to transfer UDP messages larger than limited by the IP receive MTU (EMTU_R [RFC1122]). It is typically

used with the UDP MSS option to enable more efficient use of large messages, both at the UDP and IP layers. FRAG is designed similar to the IPv6 Fragmentation Header [RFC2460], except that the UDP variant uses a 16-bit Offset measured in bytes, rather than IPv6's 13-bit Fragment Offset measured in 8-byte units. This UDP variant avoids creating reserved fields.

```
+--------+--------+--------+--------+
| Kind=8 | Len=8  |  Frag. Offset   |
+--------+--------+--------+--------+
|          Identification          |
+--------+--------+--------+--------+
```

Figure 15 UDP non-terminal FRAG option format

The FRAG option also lacks a "more" bit, zeroed for the terminal fragment of a set. This is possible because the terminal FRAG option is indicated as a longer, 12-byte variant, which includes an Internet checksum over the reassembled payload (omitting the IP pseudoheader and UDP header, as well as UDP options), as shown in Figure 16.

>> The reassembly checksum SHOULD be used, but MAY be unused in the same situations when the UDP checksum is unused (e.g., for transit tunnels or applications that have their own integrity checks [RFC2460]), and by the same mechanism (set the field to 0x0000).

```
+--------+--------+--------+--------+
| Kind=8 | Len=12 |  Frag. Offset   |
+--------+--------+--------+--------+
|          Identification          |
+--------+--------+--------+--------+
|    Checksum     |
+--------+--------+
```

Figure 16 UDP terminal FRAG option format

The Fragment Offset is 16 bits and indicates the location of the UDP payload fragment in bytes from the beginning of the original unfragmented payload. The Len field indicates whether there are more fragments (Len=8) or no more fragments (Len=12).

>> The Identification field is a 32-bit value that MUST be unique over the expected fragment reassembly timeout.

>> The Identification field SHOULD be generated in a manner similar to that of the IPv6 Fragment ID [RFC2460].

>> UDP fragments MUST NOT overlap.

FRAG needs to be used with extreme care because it will present incorrect datagram boundaries to a legacy receiver, unless encoded as LITE data (see Section 5.8.1).

>> A host SHOULD indicate FRAG support by transmitting an unfragmented datagram using the Fragmentation option (e.g., with Offset zero and length 12, i.e., including the checksum area), except when encoded as LITE.

>> A host MUST NOT transmit a UDP fragment before receiving recent confirmation from the remote host, except when FRAG is encoded as LITE.

UDP fragmentation relies on a fragment expiration timer, which can be preset or could use a value computed using the UDP Timestamp option.

>> The default UDP reassembly SHOULD be no more than 2 minutes.

Implementers are advised to limit the space available for UDP reassembly.

>> UDP reassembly space SHOULD be limited to reduce the impact of DOS attacks on resource use.

>> UDP reassembly space limits SHOULD NOT be implemented as an aggregate, to avoid cross-socketpair DOS attacks.

>> Individual UDP fragments MUST NOT be forwarded to the user. The reassembled datagram is received only after complete reassembly, checksum validation, and continued processing of the remaining options.

Any additional UDP options would follow the FRAG option in the final fragment, and would be included in the reassembled packet. Processing of those options would commence after reassembly.

>> UDP options MUST NOT follow the FRAG header in non-terminal fragments. Any data following the FRAG header in non-terminal fragments MUST be silently dropped. All other options that apply to a reassembled packet MUST follow the FRAG header in the terminal fragment.

5.8.1. Coupling FRAG with LITE

FRAG can be coupled with LITE to avoid impacting legacy receivers. Each fragment is sent as LITE un-checksummed data, where each UDP packet contains no legacy-compatible data. Legacy receivers interpret these as zero-payload packets, which would not affect the receiver unless the presence of the packet itself were a signal. The header of such a packet would appear as shown in Figure 17 and Figure 18.

```
              +---------+--------------+---------+
              | UDP Hdr |   LiteFrag   |LITE|FRAG|
              +---------+--------------+----+----+
               <-------> ^^^^           ^^^^
          Zero UDP Length  |              |
                           +--------------+
```

Figure 17 Preparing FRAG as Lite data

```
              +---------+--------------+----+
              | UDP Hdr |LITE|LiteFrag |FRAG|
              +---------+--------------+----+
               <------->  |             ^
          Zero UDP Length  |             |
                           +-------------+
```

Figure 18 Lite option before transmission

When a packet is reassembled, it appears as a complete LITE data region. The UDP header of the reassembled packet is adjusted accordingly, so that the reassembled region now appears as conventional UDP user data, and processing of the UDP options continues, as with the non-LITE FRAG variant.

5.9. Authentication and Encryption (AE)

The Authentication and Encryption option (AE) is intended to allow UDP to provide a similar type of authentication as the TCP Authentication Option (TCP-AO) [RFC5925]. It uses the same format as specified for TCP-AO, except that it uses a Kind of 8. UDP-AO supports NAT traversal in a similar manner as TCP-AO [RFC6978]. UDP-AO can also be extended to provide a similar encryption capability as TCP-AO-ENC, in a similar manner [To17ao]. For these reasons, the option is known as UDP-AE.

Like TCP-AO, UDP-AE is not negotiated in-band. Its use assumes both endpoints have populated Master Key Tuples (MKTs), used to exclude non-protected traffic.

TCP-AO generates unique traffic keys from a hash of TCP connection parameters. UDP lacks a three-way handshake to coordinate connection-specific values, such as TCP's Initial Sequence Numbers (ISNs) [RFC793], thus UDP-AE's Key Derivation Function (KDF) uses zeroes as the value for both ISNs. This means that the UDP-AE reuses keys when socket pairs are reused, unlike TCP-AO.

5.10. Experimental (EXP)

The Experimental option (EXP) is reserved for experiments [RFC3692]. Only one such value is reserved because experiments are expected to use an Experimental ID (ExIDs) to differentiate concurrent use for different purposes, using UDP ExIDs registered with IANA according to the approach developed for TCP experimental options [RFC6994].

>> The length of the experimental option MUST be at least 4 to account for the Kind, Length, and the minimum 16-bit UDP ExID identifier (similar to TCP ExIDs [RFC6994]).

6. UDP API Extensions

UDP currently specifies an application programmer interface (API), summarized as follows (with Unix-style command as an example) [RFC768]:

- Method to create new receive ports
    - E.g., bind(handle, recvaddr(optional), recvport)
- Receive, which returns data octets, source port, and source address
    - E.g., recvfrom(handle, srcaddr, srcport, data)
- Send, which specifies data, source and destination addresses, and source and destination ports
    - E.g., sendto(handle, destaddr, destport, data)

This API is extended to support options as follows:

- o Extend the method to create receive ports to include receive options that are required. Datagrams not containing these required options MUST be silently dropped and MAY be logged.

- o Extend the receive function to indicate the options and their parameters as received with the corresponding received datagram.

- o Extend the send function to indicate the options to be added to the corresponding sent datagram.

Examples of API instances for Linux and FreeBSD are provided in Appendix A, to encourage uniform cross-platform implementations.

7. Whose options are these?

UDP options are indicated in an area of the IP payload that is not used by UDP. That area is really part of the IP payload, not the UDP payload, and as such, it might be tempting to consider whether this is a generally useful approach to extending IP.

Unfortunately, the surplus area exists only for transports that include their own transport layer payload length indicator. TCP and SCTP include header length fields that already provide space for transport options by indicating the total length of the header area, such that the entire remaining area indicated in the network layer (IP) is transport payload. UDP-Lite already uses the UDP Length field to indicate the boundary between data covered by the transport checksum and data not covered, and so there is no remaining area where the length of the UDP-Lite payload as a whole can be indicated [RFC3828].

UDP options are intended for use only by the transport endpoints. They are no more (or less) appropriate to be modified in-transit than any other portion of the transport datagram.

UDP options are transport options. Generally, transport datagrams are not intended to be modified in-transit. However, the UDP option mechanism provides no specific protection against in-transit modification of the UDP header, UDP payload, or UDP option area, except as provided by the options selected (e.g., OCS, ACS, or AE).

8. UDP options vs. UDP-Lite

UDP-Lite provides partial checksum coverage, so that packets with errors in some locations can be delivered to the user [RFC3828]. It uses a different transport protocol number (136) than UDP (17) to

interpret the UDP Length field as the prefix covered by the UDP checksum.

UDP (protocol 17) already defines the UDP Length field as the limit of the UDP checksum, but by default also limits the data provided to the application as that which precedes the UDP Length. A goal of UDP-Lite is to deliver data beyond UDP Length as a default, which is why a separate transport protocol number was required.

UDP options do not need a separate transport protocol number because the data beyond the UDP Length offset (surplus data) is not provided to the application by default. That data is interpreted exclusively within the UDP transport layer.

UDP options support a similar service to UDP-Lite by terminating the UDP options with an EOL option. The additional data not covered by the UDP checksum follows that EOL option, and is passed to the user separately. The difference is that UDP-Lite provides the un-checksummed user data to the application by default, whereas UDP options can provide the same capability only for endpoints that are negotiated in advance (i.e., by default, UDP options would silently discard this non-checksummed data). Additionally, in UDP-Lite the checksummed and non-checksummed payload components are adjacent, whereas in UDP options they are separated by the option area - which, minimally, must consist of at least one EOL option.

UDP-Lite cannot support UDP options, either as proposed here or in any other form, because the entire payload of the UDP packet is already defined as user data and there is no additional field in which to indicate a separate area for options. The UDP Length field in UDP-Lite is already used to indicate the boundary between user data covered by the checksum and user data not covered.

9. Interactions with Legacy Devices

It has always been permissible for the UDP Length to be inconsistent with the IP transport payload length [RFC768]. Such inconsistency has been utilized in UDP-Lite using a different transport number. There are no known systems that use this inconsistency for UDP [RFC3828]. It is possible that such use might interact with UDP options, i.e., where legacy systems might generate UDP datagrams that appear to have UDP options. The UDP OCS provides protection against such events and is stronger than a static "magic number".

UDP options have been tested as interoperable with Linux, Max OS-X, and Windows Cygwin, and worked through NAT devices. These systems

successfully delivered only the user data indicated by the UDP Length field and silently discarded the surplus area.

One reported embedded device passes the entire IP datagram to the UDP application layer. Although this feature could enable application-layer UDP option processing, it would require that conventional UDP user applications examine only the UDP payload. This feature is also inconsistent with the UDP application interface [RFC768] [RFC1122].

It has been reported that Alcatel-Lucent’s "Brick" Intrusion Detection System has a default configuration that interprets inconsistencies between UDP Length and IP Length as an attack to be reported. Note that other firewall systems, e.g., CheckPoint, use a default "relaxed UDP length verification" to avoid falsely interpreting this inconsistency as an attack.

(TBD: test with UDP checksum offload and UDP fragmentation offload)

10. Options in a Stateless, Unreliable Transport Protocol

There are two ways to interpret options for a stateless, unreliable protocol -- an option is either local to the message or intended to affect a stream of messages in a soft-state manner. Either interpretation is valid for defined UDP options.

It is impossible to know in advance whether an endpoint supports a UDP option.

>> UDP options MUST allow for silent failure on first receipt.

>> UDP options that rely on soft-state exchange MUST allow for message reordering and loss.

>> A UDP option MUST be silently optional until confirmed by exchange with an endpoint.

The above requirements prevent using any option that cannot be safely ignored unless that capability has been negotiated with an endpoint in advance for a socket pair. Legacy systems would need to be able to interpret the transport payload fragments as individual transport datagrams.

11. UDP Option State Caching

Some TCP connection parameters, stored in the TCP Control Block, can be usefully shared either among concurrent connections or between

connections in sequence, known as TCP Sharing [RFC2140][To17cb]. Although UDP is stateless, some of the options proposed herein may have similar benefit in being shared or cached. We call this UCB Sharing, or UDP Control Block Sharing, by analogy.

[TBD: extend this section to indicate which options MAY vs. MUST NOT be shared and how, e.g., along the lines of To17cb]

Updates to RFC 768

This document updates RFC 768 as follows:

- o This document defines the meaning of the IP payload area beyond the UDP length but within the IP length.

- o This document extends the UDP API to support the use of options.

12. Security Considerations

The use of UDP packets with inconsistent IP and UDP Length fields has the potential to trigger a buffer overflow error if not properly handled, e.g., if space is allocated based on the smaller field and copying is based on the larger. However, there have been no reports of such vulnerability and it would rely on inconsistent use of the two fields for memory allocation and copying.

UDP options are not covered by DTLS (datagram transport-layer security). Despite the name, neither TLS [RFC5246] (transport layer security, for TCP) nor DTLS [RFC6347] (TLS for UDP) protect the transport layer. Both operate as a shim layer solely on the payload of transport packets, protecting only their contents. Just as TLS does not protect the TCP header or its options, DTLS does not protect the UDP header or the new options introduced by this document. Transport security is provided in TCP by the TCP Authentication Option (TCP-AO [RFC5925]) or in UDP by the Authentication Extension option (Section 5.9). Transport headers are also protected as payload when using IP security (IPsec) [RFC4301].

UDP options use the TLV syntax similar to that of TCP. This syntax is known to require serial processing and may pose a DOS risk, e.g., if an attacker adds large numbers of unknown options that must be parsed in their entirety. Implementations concerned with the potential for this vulnerability MAY implement only the required options and MAY also limit NOPs (e.g., no more than three consecutive NOPs or some total number that might occur between the required options, if all are present). Because the required options

come first and at most once each (and all later duplicates silently ignored), this limits the DOS impact.

13. IANA Considerations

Upon publication, IANA is hereby requested to create a new registry for UDP Option Kind numbers, similar to that for TCP Option Kinds. Initial values of this registry are as listed in Section 5. Additional values in this registry are to be assigned by IESG Approval or Standards Action [RFC5226].

Upon publication, IANA is hereby requested to create a new registry for UDP Experimental Option Experiment Identifiers (UDP ExIDs) for use in a similar manner as TCP ExIDs [RFC6994]. This registry is initially empty. Values in this registry are to be assigned by IANA using first-come, first-served (FCFS) rules [RFC5226].

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC768] Postel, J., "User Datagram Protocol", RFC 768, August 1980.

[RFC791] Postel, J., "Internet Protocol," RFC 791, Sept. 1981.

14.2. Informative References

[Hi15] Hildebrand, J., B. Trammel, "Substrate Protocol for User Datagrams (SPUD) Prototype," draft-hildebrand-spud-prototype-03, Mar. 2015.

[RFC793] Postel, J., "Transmission Control Protocol" RFC 793, September 1981.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -- Communication Layers," RFC 1122, Oct. 1989.

[RFC1191] Mogul, J., S. Deering, "Path MTU discovery," RFC 1191, November 1990.

[RFC1981] McCann, J., S. Deering, J. Mogul, "Path MTU Discovery for IP version 6," RFC 1981, Aug. 1996.

[RFC2140] Touch, J., "TCP Control Block Interdependence," RFC 2140, Apr. 1997.

[RFC2460] Deering, S., R. Hinden, "Internet Protocol Version 6 (IPv6) Specification," RFC 2460, Dec. 1998.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery," RFC 2923, September 2000.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, Dec. 2005.

[RFC4340] Kohler, E., M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4960] Stewart, R. (Ed.), "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful," RFC 3692, Jan. 2004.

[RFC3828] Larzon, L-A., M. Degermark, S. Pink, L-E. Jonsson (Ed.), G. Fairhurst (Ed.), "The Lightweight User Datagram Protocol (UDP-Lite)," RFC 3828, July 2004.

[RFC5226] Narten, T., H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs," RFC 5226, May 2008.

[RFC5246] Dierks, T., E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008.

[RFC5925] Touch, J., A. Mankin, R. Bonica, "The TCP Authentication Option," RFC 5925, June 2010.

[RFC6347] Rescorla, E., N. Modadugu, "Datagram Transport Layer Security Version 1.2," RFC 6347, Jan. 2012.

[RFC6691] Borman, D., "TCP Options and Maximum Segment Size (MSS)," RFC 6691, July 2012.

[RFC6978] Touch, J., "A TCP Authentication Option Extension for NAT Traversal", RFC 6978, July 2013.

[RFC6994] Touch, J., "Shared Use of Experimental TCP Options," RFC 6994, Aug. 2013.

[RFC7323] Borman, D., R. Braden, V. Jacobson, R. Scheffenegger (Ed.), "TCP Extensions for High Performance," RFC 7323, Sep. 2014.

[RFC8085] Eggert, L., G. Fairhurst, G. Shepherd, "UDP Usage Guidelines," RFC 8085, Feb. 2017.

[To17ao] Touch, J., "A TCP Authentication Option Extension for Payload Encryption", draft-touch-tcp-ao-encrypt, Apr. 2017.

[To17cb] Touch, J., M. Welzl, S. Islam, J. You, "TCP Control Block Interdependence," draft-touch-tcpm-2140bis, Jan. 2017.

[Tr15] Trammel, B. (Ed.), M. Kuelewind (Ed.), "Requirements for the design of a Substrate Protocol for User Datagrams (SPUD)," draft-trammell-spud-req-04, May 2016.

15. Acknowledgments

This work benefitted from feedback from Bob Briscoe, Ken Calvert, Ted Faber, Gorry Fairhurst, C. M. Heard (including the FRAG/LITE combination), Tom Herbert, and Mark Smith, as well as discussions on the IETF TSVWG and SPUD email lists.

This work is partly supported by USC/ISI’s Postel Center.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292 USA

Phone: +1 (310) 448-9151
Email: touch@isi.edu

Appendix A. Implementation Information

The following information is provided to encourage interoperable API implementations.

System-level variables (sysctl):

Name	default	meaning
net.ipv4.udp_opt	0	UDP options available
net.ipv4.udp_opt_ocs	1	Default include OCS
net.ipv4.udp_opt_acs	0	Default include ACS
net.ipv4.udp_opt_lite	0	Default include LITE
net.ipv4.udp_opt_mss	0	Default include MSS
net.ipv4.udp_opt_time	0	Default include TIME
net.ipv4.udp_opt_frag	0	Default include FRAG
net.ipv4.udp_opt_ae	0	Default include AE

Socket options (sockopt), cached for outgoing datagrams:

Name	meaning
UDP_OPT	Enable UDP options (at all)
UDP_OPT_OCS	Enable UDP OCS option
UDP_OPT_ACS	Enable UDP ACS option
UDP_OPT_LITE	Enable UDP LITE option
UDP_OPT_MSS	Enable UDP MSS option
UDP_OPT_TIME	Enable UDP TIME option
UDP_OPT_FRAG	Enable UDP FRAG option
UDP_OPT_AE	Enable UDP AE option

Send/sendto parameters:

(TBD - currently using cached parameters)

Connection parameters (per-socketpair cached state, part UCB):

Name	Initial value
opts_enabled	net.ipv4.udp_opt
ocs_enabled	net.ipv4.udp_opt_ocs

The following option is included for debugging purposes, and MUST NOT be enabled otherwise.

System variables

net.ipv4.udp_opt_junk 0

System-level variables (sysctl):

Name	default	meaning
net.ipv4.udp_opt_junk	0	Default use of junk

Socket options (sockopt):

Name	params	meaning
UDP_JUNK	-	Enable UDP junk option
UDP_JUNK_VAL	fillval	Value to use as junk fill
UDP_JUNK_LEN	length	Length of junk payload in bytes

Connection parameters (per-socketpair cached state, part UCB):

Name	Initial value
junk_enabled	net.ipv4.udp_opt_junk
junk_value	0xABCD
junk_len	4

Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 11, 2016

B. Trammell, Ed.
M. Kuehlewind, Ed.
ETH Zurich

May 10, 2016

Requirements for the design of a Substrate Protocol for User Datagrams (SPUD)
draft-trammell-spud-req-04

Abstract

We have identified the potential need for a UDP-based encapsulation protocol to allow explicit cooperation with middleboxes while using new, encrypted transport protocols. This document proposes an initial set of requirements for such a protocol, and discusses tradeoffs to be made in further refining these requirements.

Status of This Memo

Copyright Notice

Table of Contents

1. Motivation

A number of efforts to create new transport protocols or experiment with new network behaviors in the Internet have been built on top of UDP, as it traverses firewalls and other middleboxes more readily than new protocols do. Each such effort must, however, either manage its flows within common middlebox assumptions for UDP or train the middleboxes on the new protocol (thus losing the benefit of using UDP). A common Substrate Protocol for User Datagrams (SPUD) would allow each effort to re-use a set of shared methods for notifying middleboxes of the flows' semantics, thus avoiding both the limitations of current flow semantics and the need to re-invent the mechanism for notifying the middlebox of the new semantics.

As a concrete example, it is common for some middleboxes to tear down required state (such as NAT bindings) very rapidly for UDP flows. By notifying the path that a particular transport using UDP maintains session state and explicitly signals session start and stop using the substrate, the using protocol may reduce or avoid the need for heartbeat traffic.

The intention of this work is to make it possible to define and deploy new transport protocols that use encryption to protect their own operation as well as the confidentiality, authenticity, integrity, and linkability resistance of their payloads. The accelerating deployment of encryption will render obsolete network operations techniques that rely on packet inspection and modification based upon assumptions about the protocols in use. This work will allow the replacement the current regime of middlebox inspection and modification of transport and application-layer headers and payload with one that allows inspection only of information explicitly exposed by the endpoints, and modification of such information only under endpoint control.

Any selective exposure of traffic metadata outside a relatively restricted trust domain must be advisory, non-negotiated, and declarative rather than imperative. As with other signaling systems, exposure of specific elements must be carefully assessed for privacy risks and the total of exposed elements must be so assessed. Each exposed parameter should also be independently verifiable, so that each entity can assign its own trust to other entities. Basic transport over the substrate must continue working even if signaling is ignored or stripped, to support incremental deployment. These restrictions on vocabulary are discussed further in [I-D.trammell-stackevo-explicit-coop]. This discussion includes privacy and trust concerns as well as the need for strong incentives for middlebox cooperation based on the information that are exposed.

Within this document, requirements are presented for a facility implementable as an encapsulation protocol, atop which new transports ("superstrates") can be built. Alternately, these could be viewed as a set of requirements for future transport protocol development without a layer separation between the transport and the superstrate.

This document defines a specific set of requirements for a SPUD facility, based on analysis on a target set of applications. It is intended as the basis for determining the next steps to make progress in this space, including possibly chartering a working group for specific protocol engineering work.

2. History

An outcome of the IAB workshop on Stack Evolution in a Middlebox Internet (SEMI) [RFC7663], held in Zurich in January 2015, was a discussion on the creation of a substrate protocol to support the deployment of new transport protocols in the Internet. Assuming that a way forward for transport evolution in user space would involve encapsulation in UDP datagrams, the workshop noted that it may be useful to have a facility built atop UDP to provide minimal signaling of the semantics of a flow that would otherwise be available in TCP. At the very least, indications of first and last packets in a flow may assist firewalls and NATs in policy decision and state maintenance. This facility could also provide minimal application-to- path and path-to-application signaling, though there was less agreement about what should or could be signaled here. Further transport semantics would be used by the protocol running atop this facility, but would only be visible to the endpoints, as the transport protocol headers themselves would be encrypted, along with the payload, to prevent inspection or modification. This encryption might be accomplished by using DTLS [RFC6347] as a subtransport [I-D.huitema-tls-dtls-as-subtransport] or by other suitable methods.

The Substrate Protocol for User Datagrams (SPUD) BoF was held at IETF 92 in Dallas in March 2015 to develop this concept further. Restrictions on vocabulary assumed in these requirements are derived from discussions during this BoF, based on experience with previous endpoint-to-middle and middle-to- endpoint signaling approaches as well as concerns about the privacy implications of endpoint-to-middle signaling.

3. Terminology

This document uses the following terms:

- o Superstrate: The transport protocol or protocol stack "above" SPUD, that uses SPUD for explicit path cooperation and path

traversal. The superstrate usually consists of a security layer (e.g. TLS, DTLS) and a transport protocol, or a transport protocol with integrated security features, to protect headers and payload above SPUD.

- o Endpoint: One end of a communication session, located on a single node that is a source or destination of packets in that session. In this document, this term may refer to either the SPUD implementation at the endpoint, the superstrate implementation running over SPUD, or the applications running over that superstrate.

- o Path: The sequence of Internet Protocol nodes and links that a given packet traverses from endpoint to endpoint.

- o Middlebox: As defined in [RFC3234], a middlebox is any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host; e.g. making decisions about forwarding behavior based on other than addressing information, and/or modifying a packet before forwarding.

4. Use Cases

Use cases are outlined in more detail in [I-D.kuehlewind-spud-use-cases]. We summarize some of the primary use cases below.

The primary use case for endpoint to path signaling in the Internet making use of packet grouping, as described in the use case document, is the binding of limited related semantics (start, ack, and stop) to a flow or a group of packets within a flow that are semantically related in terms of the application or superstrate. By explicitly signaling start and stop semantics, a flow allows middleboxes to use those signals for setting up and tearing down their relevant state (NAT bindings, firewall pinholes), rather than requiring the middlebox to infer this state from continued traffic. At best, this would allow the application to reduce heartbeat traffic, which might result in reduced radio utilization and thus greater battery life on mobile platforms.

SPUD could also be used to provide information relevant for network treatment for middleboxes as a replacement for deep packet inspection for traffic classification purposes, rendered ineffective by superstrate encryption. In this application, properties would be expressed in terms of network-relevant parameters (intended bandwidth, latency and loss sensitivity, etc.) as opposed to application-relevant semantics. See

[I-D.trammell-stackevo-explicit-coop] for discussion on limitations in signaling in untrusted environments.

SPUD may also provide some facility for SPUD-aware nodes on the path to signal some property of the path to the endpoints and other SPUD-aware nodes on the path. The primary use case for path to application signaling is parallel to the use of ICMP [RFC0792] and ICMPv6 [RFC4443], in that it describes a set of conditions (including errors) that applies to the datagrams as they traverse the path. Since the signals here would traverse NATs in the same way as the traffic related to them, this use case would sidestep problems with ICMP availability in the deployed Internet.

Link-layer characteristics of use to the transport layer (e.g., whether a high-transient-delay, highly-buffered link such as LTE is present on the path) could also be signaled using this path-to-endpoint facility.

5. Functional Requirements

The following requirements detail the services that SPUD must provide to superstrates, endpoints, and middleboxes using SPUD.

5.1. Grouping of Packets (into "tubes")

Transport semantics and many properties of communication that endpoints may want to expose to middleboxes are bound to flows or groups of flows (5-tuples). SPUD must therefore provide a basic facility for associating packets together (into what we call a "tube", for lack of a better term) and associate information to these groups of packets. Each packet in a SPUD "flow" (determined by 5-tuple) belongs to exactly one tube. Notionally, a tube consists of a set of packets with a set of common properties, that should therefore receive equivalent treatment from the network; these tubes may or may not be related to separate semantic entities in the superstrate (e.g. SCTP streams), at the superstrate’s discretion.

The simplest mechanisms for association involve the addition of an identifier to each packet in a tube. Other mechanisms that don’t directly encode the identifier in a packet header, but instead provide it in a way that it is simple to derive from other information available in the packet at the endpoints and along the path, are also possible. In any cases, for the purposes of this requirement we treat this identifier as a simple vector of N bits. The properties of the tube identifier are subject to tradeoffs on the requirements for privacy, security, ease of implementation, and header overhead efficiency.

In determining the optimal size and scope for this tube identifier, we first assume that the 5-tuple of source and destination IP address, UDP port, and IP protocol identifier (17 for UDP) is used in the Internet as an existing flow identifier, due to the widespread deployment of network address and port translation. We conclude that SPUD tube IDs should be scoped to this 5-tuple.

While a globally-unique identifier would allow easier state comparison and migration for mobility use cases, it would have two serious disadvantages. First, N would need to be sufficiently large to minimize the probability of collision among multiple tubes having the same identifier along the same path during some period of time. A 128-bit UUID [RFC4122] or an identifier of equivalent size generated using an equivalent algorithm would probably be sufficient, at the cost of 128 bits of header space in every packet. Second, globally unique tube identifiers would also introduce new possibilities for user and node tracking, with a serious negative impact on privacy. We note that global identifiers for mobility, when necessary to expose to the path, can be supported separately from the tube identification mechanism, by using a generic tube-grouping application-to-path signaling bound to the tube.

Even when tube IDs are scoped to 5-tuples, N must still be sufficiently large, and the bits in the identifier sufficiently random, that possession of a valid tube ID implies that a node can observe packets belonging to the tube. This reduces the chances of success of blind packet injection attacks of packets with guessed valid tube IDs.

5.2. Bidirectionality of Tubes

When scoped to 5-tuples, the forward and backward directions of a bidirectional connection will have different tube IDs, since these will necessarily take different paths and may interact with a different set of middleboxes due to asymmetric routing. SPUD will therefore require some facility to note that one tube is the "reverse" direction of another, a general case of the tube grouping signal above.

5.3. Signaling of Per-Tube Properties

SPUD must be able to provide information scoped to a tube from the end- point(s) to all SPUD-aware nodes on the path about the packets in that tube.

We note that in-band signaling would meet this requirement.

5.4. Path to Receiver Signaling under Sender Control

SPUD must be able to provide information about from a SPUD-aware middlebox to the endpoint. This information is associated with a tube, in terms of "the properties of the path(s) the packets in this tube will traverse". This signaling must happen only with explicit sender permission and be sent to the receiver of packets in the tube.

We note that in-band signaling would meet this requirement, if the sender created a "placeholder" in-band that could be filled in by the middlebox(es) on path. In-band signaling has the advantage that it does not require foreknowledge of the identity and addresses of devices along the path by endpoints and vice versa, but does add complexity to the signaling protocol. Piggybacked signaling uses some number of bits in each packet generated by the overlying transport. It requires either reducing the MTU available to the encapsulated transport and/or opportunistically using "headroom" as it is available: bits between the network-layer MTU and the bits actually used by the transport. For use cases that accumulate information from devices on path in the SPUD header, piggybacked signaling also requires a mechanism for endpoints to create "scratch space" for potential use of the on-path devices.

In contrast, interleaved signaling uses signaling packets on the same 5-tuple and tube ID, which don’t carry any superstrate data. These interleaved packets could also contain scratch space for on-path device use. This reduces complexity and sidesteps MTU problems, at the cost of sending more packets per flow.

5.5. Receiver to Sender Feedback

SPUD must be able send information collected from SPUD-aware middleboxes along the path to a receiver back to the sender that gave permission; see Section 6.4 for restrictions on this facility.

5.6. Direct Path to Sender Signaling

SPUD must provide a facility for a middlebox to send a packet directly in response to a sending endpoint, primarily to signal error conditions (e.g. "packet administratively prohibited" or "no route to destination", as in present ICMP).

In this case, the direct return packet generated by the middlebox uses the reversed end-to-end 5-tuple in order to receive equivalent NAT treatment, though the reverse path might not be the same as the forward path. Endpoints have control over this feature: A SPUD-aware middlebox must not emit a direct return packet unless it is in direct response to a packet from a sending endpoint, and must not forward a

packet for which it has sent a direct return packet; see Section 6.6 and Section 7.9.

5.7. Tube Start and End Signaling

SPUD must provide a facility for endpoints to signal that a tube has started, that the start of the tube has been acknowledged and accepted by the remote endpoint(s), and that a tube has ended and its state can be forgotten by the path. Given unreliable signaling (see Section 7.10), both endpoints and devices on the path must be resilient to the loss of any of these signals. Specifically, timeouts are still necessary to clean up stale state.

5.8. Transport Semantic Signaling

Similar to tube start and end signaling, SPUD must provide a facility for endpoints to signal that a superstrate transport session has been requested, set up, and/or torn down. This facility provides an explicit replacement for the common practice in TCP-aware middleboxes of modeling TCP state of flows by inspecting the TCP flags byte.

Given the fact that a superstrate transport session may consist of multiple tubes, this signaling must be separate from that for tube start and end.

5.9. Declarative signaling

All information signaled via SPUD is defined to be declarative (as opposed to imperative). A SPUD endpoint must function correctly even no middlebox along the path understands the signals it sends, or if sent signals from middleboxes it does not understand. It must also function correctly if the path (and thereby the set of middleboxes traversed) changes during the lifetime of a tube; endpoints cannot rely on the creation or maintenance of state even on cooperative middleboxes. Likewise, a SPUD-aware middlebox must function correctly if sent signals from endpoints it does not understand, or in the absence of expected signals from endpoints.

The declarative nature of this signaling removes any requirement that SPUD provide reliability for its signals.

5.10. Extensibility

SPUD must enable multiple new transport semantics and application/path declarations without requiring updates to SPUD implementations in middleboxes.

The use of SPUD for experimental signaling must be possible either without the registration of codepoints or namespaces with IANA, or with trivially easy (First Come, First Served [RFC5226] registration of such codepoints.

5.11. Common Vocabulary

For the interoperability of SPUD endpoints and middleboxes with each other, the use of SPUD for standard signaling must use a common vocabulary with registered codepoints allocated under relatively restrictive policy. This restrictive policy serves primarily security and privacy goals (i.e., reducing the risk of misuse of the extensibility provided by the protocol).

We note that an IANA registry requiring Standards Action {RFC5226}} to modify would meet this requirement.

5.12. Additional Per-Packet Signaling

SPUD may provide a facility for signaling semantically simple information (similar to tube start and end) on a per-packet as opposed to a per-tube basis. Properties signaled per packet reduce state requirements at middleboxes, but also increase per-packet overhead. Small signal size (in bits of entropy) and encoding efficiency (in bits on the wire) is therefore more important for per-packet signaling that per-tube signaling. If per-packet signals need to be used by multiple hops along a path, these will need to be encoded in an efficiently-implementable way (i.e., using fixed-length, constant-offset data structures).

Given these constraints, per-packet signaling is necessary for certain use cases, it is likely that SPUD will provide a very limited set of per-packet signals using flags in a SPUD header, and require all more complex properties to be bound per-tube.

6. Security Requirements

6.1. Privacy

SPUD must allow endpoints to control the amount of information exposed to middleboxes, with the default being the minimum necessary for correct functioning. This includes the cryptographic protection of transport layer headers from inspection by devices on path, in order to prevent ossification of these headers.

6.2. Authentication

The basic SPUD protocol must not require any authentication or a priori trust relationship between endpoints and middleboxes to function. However, SPUD should interoperate with the presentation/exchange of authentication information in environments where a trust relationship already exists, or can be easily established, either in-band or out-of-band, and use this information where possible and appropriate.

Given the advisory nature of the signaling it supports, SPUD may also support eventual authentication: authentication of a signal after the reception of a packet after that containing the signal.

6.3. Integrity

SPUD must be able to provide integrity protection of information exposed by endpoints in SPUD-encapsulated packets, though the details of this integrity protection are still open.

Endpoints should be able to detect changes to headers SPUD uses for its own signaling (whether due to error, accidental modification, or malicious modification), as well as the injection of packets into a SPUD flow (defined by 5-tuple) or tube by nodes other than the remote endpoints. Errors and accidental modifications can be detected using a simple checksum over the SPUD header, while detecting malicious modifications requires cryptographic integrity protection. Similar to Section 6.2, cryptographic integrity protection may also be eventual.

Integrity protection of the superstrate is left up to the superstrate.

6.4. Encrypted Feedback

As feedback from a receiver to a sender (see Section 5.5) does not need to be exposed to the path, this feedback channel should be encrypted for confidentiality and authenticity, when available (see Section 6.2). This facility will rely on cooperation with the superstrate or some other out-of-band mechanism to provide these guarantees.

6.5. Preservation of Security Properties

The use of SPUD must not weaken the essential security properties of the superstrate: confidentiality, integrity, authenticity, and defense against linkability. If the superstrate includes payload encryption for confidentiality, for example, the use of SPUD must not

allow deep packet inspection systems to have access to the plaintext. Likewise, the use of SPUD must not create additional opportunities for linkability not already existing in the superstrate.

6.6. Protection against trivial abuse

Malicious background traffic is a serious problem for UDP-based protocols due to the ease of forging source addresses in UDP together with only limited deployment of network egress filtering [RFC2827]. Trivial abuse includes flooding and state exhaustion attacks, as well as reflection and amplification attacks. SPUD must provide minimal protection against this trivial abuse. This implies that SPUD should provide:

- o a proof of return routability, that the endpoint identified by a packet’s source address receives packets sent to that address;
- o a feedback channel between endpoints;
- o a method to probabilistically discriminiate legitimate SPUD traffic from reflected malicious traffic;
- o a method to probabilistically discriminate SPUD traffic from on-path devices from devices off-path; and
- o the ability to deploy mechanisms to protect against state exhaustion and other denial-of-service attacks against SPUD itself.

We note that using a "magic number" or other pattern of bits in an encapsulation-layer header not used in any widely deployed protocol has the nice property that no existing node in the Internet can be induced to reflect traffic containing it. This allows the magic number to provide probabilistic assurance that a given packet is not reflected, assisting in meeting this requirement.

If SPUD is implemented over UDP, see [I-D.ietf-tsvwg-rfc5405bis] for guidelines on the safe usage of UDP in the Internet, which addresses some of these issues.

6.7. Continuum of trust among endpoints and middleboxes

There are different security considerations for different security contexts. The end-to-end context is one; anything that only needs to be seen by the path shouldn’t be exposed in SPUD, but rather by the superstrate. There are multiple different types of end-to-middle context based on levels of trust between end and middle - is the middlebox on the same network as the endpoint, under control of the

same owner? Is there some contract between the application user and the middlebox operator? SPUD should support different levels of trust than the default ("untrusted, but presumed honest due to limitations on the signaling vocabulary") and fully-authenticated; how these points along the continuum are to be implemented and how they relate to each other needs to be explored further.

In the Internet, it is not in the general case possible for the endpoint to authenticate every middlebox that might see packets it sends and receives. In this case information produced by middleboxes may enjoy less integrity protection than that produced by endpoints. In addition, endpoint authentication of middleboxes and vice-versa may be better conducted out-of- band (treating the middlebox as an endpoint for the authentication protocol) than in-band (treating the middlebox as a participant in a 3+ party communication).

7. Technical Requirements

The following requirements detail the constraints on how the SPUD facility must meet its functional requirements.

7.1. Middlebox Traversal

SPUD, including all path-to-endpoint and endpoint-to-path signaling as well as superstrate and superstrate payload, should be able to traverse existing middleboxes and firewalls, including those that are not SPUD-aware. Therefore SPUD must be encapsulated in a transport protocol that is known to be accepted on a large fraction of paths in the Internet, or implement some form of probing to determine in advance which transport protocols will be accepted on a certain path. This encapsulation will require port numbers to support endpoints connected via network address and port translation (NAPT). We note that UDP encapsulation would meet these requirements.

7.2. Low Overhead in Network Processing

SPUD must be desgined to have low overhead, specifically requiring very little effort to recognize that a packet is a SPUD packet and to determine the tube it is associated with. We note that a magic number as in
Section 6.6 would also have a low probability of colliding with any non-SPUD traffic, therefore meeting the recognition requirement. Tube identifiers appearing directly in the encapsulation-layer header would meet the tube association requirement.

7.3. Implementability in User-Space

To enable fast deployment SPUD and superstrates must be implementable without requiring kernel replacements or modules on the endpoints, and without having special privilege (such as is required for raw packet transmission, i.e. root or "jailbreak") on the endpoints.

We note here that UDP would meet this requirement, as nearly all operating systems and application development platforms allow a userspace application to open UDP sockets.

We additionally note that while TCP APIs are also widely available to userspace applications, they are bound to TCP transport semantics, and generally do not provide enough control over segmentation and transmission to successfully implement superstrate transports.

7.4. Incremental Deployability

SPUD must be designed to operate in the present Internet, and must be designed to encourage incremental deployment.

As endpoint implementations can change more quickly than middleboxes can be designed and deployed, a SPUD facility that was be useful between endpoints even before the deployment of middleboxes that understand it would stimulate deployment. The information exposed over SPUD must provide incentives for adoption by both endpoints and middleboxes.

SPUD must not be designed in such a way that precludes its deployability in multipath, multicast, and/or endpoint multi-homing environments.

7.5. No unnecessary restrictions on the superstrate

Beyond those restrictions deemed necessary as common features of any secure, responsible transport protocol (see Section 6.6), SPUD must impose only minimal restrictions on the transport protocols it encapsulates. However, to serve as a substrate, it is necessary to factor out the information that middleboxes commonly rely on and endpoints are commonly willing to expose. This information should be included in SPUD, and might itself impose additional restrictions to the superstrate.

7.6. Minimal additional start-up latency

SPUD should not introduce additional start-up latency for superstrates. Specifically, superstrates which can send data on an initial packet must be able to do so when encapsulated within SPUD.

7.7. Minimal header overhead

To avoid reducing network performance, the information and coding used in SPUD should be designed to use the minimum necessary amount of additional space in encapsulation headers.

7.8. Minimal non-productive traffic

SPUD should minimize additional non-productive traffic (e.g. keepalives), and should provide mechanisms to allow its superstrates to minimize their reliance on non-productive traffic.

7.9. Endpoint Control

Both endpoint-to-path and path-to-endpoint signaling happen completely under endpoint control.

7.10. On Reliability, Fragmentation, MTU, and Duplication

As any information provided by SPUD is anyway opportunistic, SPUD need not provide reliable signaling for the information associated with a tube. Signals must be idempotent; all middleboxes and endpoints must gracefully handle receiving duplicate signal information. SPUD must continue working in the presence of IPv4 fragmentation on path, but in order to reduce the impact of requiring fragments reassembly at middleboxes for signals to be intelligible, endpoints using SPUD should attempt to fit all signals into single MTU-sized packets.

Given the importance of good path MTU information to SPUD’s own signaling, SPUD should implement packetization layer path MTU discovery [RFC4821].

Any facilities requiring more than an MTU’s worth of data in a single signal should use an out-of-band method which does provide reliability - this method may be an existing transport or superstrate/SPUD combination, or a "minimal transport" defined by SPUD for its own use.

7.11. SPUD Support Discovery

If SPUD is not usable on a path to an endpoint, a SPUD sender needs to be able to fall back to some other approach to achieve the goals of the superstrate; a SPUD endpoint must be able to easily determine whether a remote endpoint with which it wants to communicate using SPUD as a substrate can support SPUD, and whether path to the remote endpoint as well as the return path from the remote endpoint will pass SPUD packets.

It is not clear whether this is a requirement of SPUD, or a requirement of the superstrate / application over SPUD.

8. Security Considerations

The security-relevant requirements for SPUD are outlined in Section 6. These will be further addressed in protocol definition work following from these requirements.

9. IANA Considerations

This document has no actions for IANA.

10. Contributors

In addition to the editors, this document is the work of David Black, Ken Calvert, Ted Hardie, Joe Hildebrand, Jana Iyengar, and Eric Rescorla.

11. Acknowledgments

Thanks to Ozgu Alay, Roland Bless, Cameron Byrne, Toerless Eckert, Gorry Fairhurst, Daniel Kahn Gillmor, Tom Herbert, Christian Huitema, Iain Learmonth, Diego Lopez, and Matteo Varvello for feedback and comments on these requirements, as well as to the participants at the SPUD BoF at IETF 92 meeting in Dallas and the IAB SEMI workshop in Zurich for the discussions leading to this work.

This work is supported by the European Commission under Horizon 2020 grant agreement no. 688421 Measurement and Architecture for a Middleboxed Internet (MAMI), and by the Swiss State Secretariat for Education, Research, and Innovation under contract no. 15.0268. This support does not imply endorsement.

12. Informative References

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <http://www.rfc-editor.org/info/rfc792>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <http://www.rfc-editor.org/info/rfc3234>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <http://www.rfc-editor.org/info/rfc4122>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <http://www.rfc-editor.org/info/rfc4821>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <http://www.rfc-editor.org/info/rfc7510>.

[RFC7663] Trammell, B., Ed. and M. Kuehlewind, Ed., "Report from the IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI)", RFC 7663, DOI 10.17487/RFC7663, October 2015, <http://www.rfc-editor.org/info/rfc7663>.

[I-D.ietf-tsvwg-rfc5405bis] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", draft-ietf-tsvwg-rfc5405bis-11 (work in progress), April 2016.

[I-D.kuehlewind-spud-use-cases] KĂźhlewind, M. and B. Trammell, "Use Cases for a Substrate Protocol for User Datagrams (SPUD)", draft-kuehlewind-spud-use-cases-01 (work in progress), March 2016.

[I-D.huitema-tls-dtls-as-subtransport]
Huitema, C., Rescorla, E., and J. Jana, "DTLS as Subtransport protocol", draft-huitema-tls-dtls-as-subtransport-00 (work in progress), March 2015.

[I-D.trammell-stackevo-explicit-coop]
Trammell, B., "Architectural Considerations for Transport Evolution with Explicit Path Cooperation", draft-trammell-stackevo-explicit-coop-00 (work in progress), September 2015.

Authors' Addresses

Brian Trammell (editor)
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: ietf@trammell.ch

Mirja Kuehlewind (editor)
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: mirja.kuehlewind@tik.ee.ethz.ch

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

R. Stewart
Netflix, Inc.
M. Tuexen
Muenster Univ. of Appl. Sciences
M. Proshin
Ericsson
July 8, 2016

RFC 4960 Errata and Issues
draft-tuexen-tsvwg-rfc4960-errata-04.txt

Abstract

This document is a compilation of issues found since the publication of RFC4960 in September 2007 based on experience with implementing, testing, and using SCTP along with the suggested fixes. This document provides deltas to RFC4960 and is organized in a time based way. The issues are listed in the order they were brought up. Because some text is changed several times the last delta in the text is the one which should be applied. In addition to the delta a description of the problem and the details of the solution are also provided.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2017.

Table of Contents

1. Introduction

This document contains a compilation of all defects found up until the publishing of this document for [RFC4960] specifying the Stream Control Transmission Protocol (SCTP). These defects may be of an editorial or technical nature. This document may be thought of as a companion document to be used in the implementation of SCTP to clarify errors in the original SCTP document.

This document provides a history of the changes that will be compiled into a BIS document for [RFC4960]. It is structured similar to [RFC4460].

Each error will be detailed within this document in the form of:

- o The problem description,
- o The text quoted from [RFC4960],
- o The replacement text that should be placed into an upcoming BIS document,
- o A description of the solution.

Note that when reading this document one must use care to assure that a field or item is not updated further on within the document. Each section should be applied in sequence to the original [RFC4960] since this document is a historical record of the sequential changes that have been found necessary at various inter-op events and through discussion on the list.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Corrections to RFC 4960

3.1. Path Error Counter Threshold Handling

3.1.1. Description of the Problem

The handling of the 'Path.Max.Retrans' parameter is described in Section 8.2 and Section 8.3 of [RFC4960] in an Inconsistent way. Whereas Section 8.2 describes that a path is marked inactive when the path error counter exceeds the threshold, Section 8.3 says the path is marked inactive when the path error counter reaches the threshold.

This issue was reported as an Errata for [RFC4960] with Errata ID 1440.

3.1.2. Text Changes to the Document

```
---------
Old text: (Section 8.3)
---------
```

When the value of this counter reaches the protocol parameter 'Path.Max.Retrans', the endpoint should mark the corresponding destination address as inactive if it is not so marked, and may also optionally report to the upper layer the change of reachability of this destination address. After this, the endpoint should continue HEARTBEAT on this destination address but should stop increasing the counter.

```
---------
New text: (Section 8.3)
---------
```

When the value of this counter exceeds the protocol parameter 'Path.Max.Retrans', the endpoint should mark the corresponding destination address as inactive if it is not so marked, and may also optionally report to the upper layer the change of reachability of this destination address. After this, the endpoint should continue HEARTBEAT on this destination address but should stop increasing the counter.

3.1.3. Solution Description

The intended state change should happen when the threshold is exceeded.

3.2. Upper Layer Protocol Shutdown Request Handling

3.2.1. Description of the Problem

Section 9.2 of [RFC4960] describes the handling of received SHUTDOWN chunks in the SHUTDOWN-RECEIVED state instead of the handling of shutdown requests from its upper layer in this state.

This issue was reported as an Errata for [RFC4960] with Errata ID 1574.

3.2.2. Text Changes to the Document

```
---------
Old text: (Section 9.2)
---------
```

Once an endpoint has reached the SHUTDOWN-RECEIVED state, it MUST NOT send a SHUTDOWN in response to a ULP request, and should discard subsequent SHUTDOWN chunks.

```
---------
New text: (Section 9.2)
---------
```

Once an endpoint has reached the SHUTDOWN-RECEIVED state, it MUST NOT send a SHUTDOWN in response to a ULP request, and should discard subsequent ULP shutdown requests.

3.2.3. Solution Description

The text never intended the SCTP endpoint to ignore SHUTDOWN chunks from its peer. If it did the endpoints could never gracefully terminate associations in some cases.

3.3. Registration of New Chunk Types

3.3.1. Description of the Problem

Section 14.1 of [RFC4960] should deal with new chunk types, however, the text refers to parameter types.

This issue was reported as an Errata for [RFC4960] with Errata ID 2592.

3.3.2. Text Changes to the Document

```
---------
Old text: (Section 14.1)
---------

The assignment of new chunk parameter type codes is done through an
IETF Consensus action, as defined in [RFC2434].  Documentation of the
chunk parameter MUST contain the following information:

---------
New text: (Section 14.1)
---------

The assignment of new chunk type codes is done through an
IETF Consensus action, as defined in [RFC2434].  Documentation of the
chunk type MUST contain the following information:
```

3.3.3. Solution Description

Refer to chunk types as intended.

3.4. Variable Parameters for INIT Chunks

3.4.1. Description of the Problem

Newlines in wrong places break the layout of the table of variable parameters for the INIT chunk in Section 3.3.2 of [RFC4960].

This issue was reported as an Errata for [RFC4960] with Errata ID 3291 and Errata ID 3804.

3.4.2. Text Changes to the Document

```
---------
Old text: (Section 3.3.2)
---------

Variable Parameters                  Status     Type Value
-----------------------------------------------------------------
IPv4 Address (Note 1)               Optional    5 IPv6 Address
(Note 1)               Optional    6 Cookie Preservative
Optional    9 Reserved for ECN Capable (Note 2)   Optional
32768 (0x8000) Host Name Address (Note 3)          Optional
11 Supported Address Types (Note 4)    Optional    12

---------
New text: (Section 3.3.2)
---------

Variable Parameters                  Status     Type Value
-----------------------------------------------------------------
IPv4 Address (Note 1)               Optional    5
IPv6 Address (Note 1)               Optional    6
Cookie Preservative                 Optional    9
Reserved for ECN Capable (Note 2)   Optional    32768 (0x8000)
Host Name Address (Note 3)          Optional    11
Supported Address Types (Note 4)    Optional    12
```

3.4.3. Solution Description

Fix the formatting of the table.

3.5. CRC32c Sample Code on 64-bit Platforms

3.5.1. Description of the Problem

The sample code for computing the CRC32c provided in [RFC4960] assumes that a variable of type unsigned long uses 32 bits. This is not true on some 64-bit platforms (for example the ones using LP64).

This issue was reported as an Errata for [RFC4960] with Errata ID 3423.

3.5.2. Text Changes to the Document

```
---------
Old text: (Appendix C)
---------

unsigned long
generate_crc32c(unsigned char *buffer, unsigned int length)
{
  unsigned int i;
  unsigned long crc32 = ~0L;

---------
New text: (Appendix C)
---------

unsigned long
generate_crc32c(unsigned char *buffer, unsigned int length)
{
  unsigned int i;
  unsigned long crc32 = 0xffffffffL;
```

3.5.3. Solution Description

Use 0xffffffffL instead of ~0L which gives the same value on platforms using 32 bits or 64 bits for variables of type unsigned long.

3.6. Endpoint Failure Detection

3.6.1. Description of the Problem

The handling of the association error counter defined in Section 8.1 of [RFC4960] can result in an association failure even if the path used for data transmission is available, but idle.

This issue was reported as an Errata for [RFC4960] with Errata ID 3788.

3.6.2. Text Changes to the Document

```
---------
Old text: (Section 8.1)
---------

An endpoint shall keep a counter on the total number of consecutive
retransmissions to its peer (this includes retransmissions to all the
destination transport addresses of the peer if it is multi-homed),
including unacknowledged HEARTBEAT chunks.

---------
New text: (Section 8.1)
---------

An endpoint shall keep a counter on the total number of consecutive
retransmissions to its peer (this includes data retransmissions
to all the destination transport addresses of the peer if it is
multi-homed), including the number of unacknowledged HEARTBEAT
chunks observed on the path which currently is used for data
transfer. Unacknowledged HEARTBEAT chunks observed on paths
different from the path currently used for data transfer shall
not increment the association error counter, as this could lead
to association closure even if the path which currently is used for
data transfer is available (but idle).
```

3.6.3. Solution Description

A more refined handling for the association error counter is defined.

3.7. Data Transmission Rules

3.7.1. Description of the Problem

When integrating the changes to Section 6.1 A) of [RFC2960] as described in Section 2.15.2 of [RFC4460] some text was duplicated and became the final paragraph of Section 6.1 A) of [RFC4960].

This issue was reported as an Errata for [RFC4960] with Errata ID 4071.

3.7.2. Text Changes to the Document

```
---------
Old text: (Section 6.1 A))
---------
The sender MUST also have an algorithm for sending new DATA chunks
to avoid silly window syndrome (SWS) as described in [RFC0813].
The algorithm can be similar to the one described in Section
4.2.3.4 of [RFC1122].

However, regardless of the value of rwnd (including if it is 0),
the data sender can always have one DATA chunk in flight to the
receiver if allowed by cwnd (see rule B below).  This rule allows
the sender to probe for a change in rwnd that the sender missed
due to the SACK having been lost in transit from the data receiver
to the data sender.

---------
New text: (Section 6.1 A))
---------

The sender MUST also have an algorithm for sending new DATA chunks
to avoid silly window syndrome (SWS) as described in [RFC0813].
The algorithm can be similar to the one described in Section
4.2.3.4 of [RFC1122].
```

3.7.3. Solution Description

Last paragraph of Section 6.1 A) removed as intended in Section 2.15.2 of [RFC4460].

3.8. T1-Cookie Timer

3.8.1. Description of the Problem

Figure 4 of [RFC4960] illustrates the SCTP association setup. However, it incorrectly shows that the T1-init timer is used in the COOKIE-ECHOED state whereas the T1-cookie timer should have been used instead.

This issue was reported as an Errata for [RFC4960] with Errata ID 4400.

3.8.2. Text Changes to the Document

```
---------
Old text: (Section 5.1.6, Figure 4)
---------

COOKIE ECHO [Cookie_Z] ------\
(Start T1-init timer)         \
(Enter COOKIE-ECHOED state)    \---> (build TCB enter ESTABLISHED
                                      state)
                               /---- COOKIE-ACK
                              /
(Cancel T1-init timer, <-----/
 Enter ESTABLISHED state)

---------
New text: (Section 5.1.6, Figure 4)
---------

COOKIE ECHO [Cookie_Z] ------\
(Start T1-cookie timer)       \
(Enter COOKIE-ECHOED state)    \---> (build TCB enter ESTABLISHED
                                      state)
                               /---- COOKIE-ACK
                              /
(Cancel T1-cookie timer, <---/
 Enter ESTABLISHED state)
```

3.8.3. Solution Description

Change the figure such that the T1-cookie timer is used instead of the T1-init timer.

3.9. Miscellaneous Typos

3.9.1. Description of the Problem

While processing [RFC4960] some typos were not catched.

3.9.2. Text Changes to the Document

```
---------
Old text: (Section 1.6)
---------

Transmission Sequence Numbers wrap around when they reach 2**32 - 1.
That is, the next TSN a DATA chunk MUST use after transmitting TSN =
2*32 - 1 is TSN = 0.

---------
New text: (Section 1.6)
---------

Transmission Sequence Numbers wrap around when they reach 2**32 - 1.
That is, the next TSN a DATA chunk MUST use after transmitting TSN =
2**32 - 1 is TSN = 0.

---------
Old text: (Section 3.3.10.9)
---------

No User Data: This error cause is returned to the originator of a

DATA chunk if a received DATA chunk has no user data.

---------
New text: (Section 3.3.10.9)
---------

No User Data: This error cause is returned to the originator of a
DATA chunk if a received DATA chunk has no user data.
```

```
---------
Old text: (Section 6.7, Figure 9)
---------

Endpoint A                                      Endpoint Z {App
sends 3 messages; strm 0} DATA [TSN=6,Strm=0,Seq=2] ----------
-----> (ack delayed) (Start T3-rtx timer)

DATA [TSN=7,Strm=0,Seq=3] --------> X (lost)

DATA [TSN=8,Strm=0,Seq=4] ---------------> (gap detected,
                                            immediately send ack)
                                 /----- SACK [TSN Ack=6,Block=1,
                                /             Start=2,End=2]
                         <-----/ (remove 6 from out-queue,
 and mark 7 as "1" missing report)

---------
New text: (Section 6.7, Figure 9)
---------

Endpoint A                                      Endpoint Z
{App sends 3 messages; strm 0}
DATA [TSN=6,Strm=0,Seq=2] ---------------> (ack delayed)
(Start T3-rtx timer)

DATA [TSN=7,Strm=0,Seq=3] --------> X (lost)

DATA [TSN=8,Strm=0,Seq=4] ---------------> (gap detected,
                                            immediately send ack)
                                 /----- SACK [TSN Ack=6,Block=1,
                                /             Strt=2,End=2]
                         <-----/
(remove 6 from out-queue,
 and mark 7 as "1" missing report)
```

```
---------
Old text: (Section 6.10)
---------
An endpoint bundles chunks by simply including multiple chunks in one
outbound SCTP packet.  The total size of the resultant IP datagram,

including the SCTP packet and IP headers, MUST be less that or equal
to the current Path MTU.

---------
New text: (Section 6.10)
---------

An endpoint bundles chunks by simply including multiple chunks in one
outbound SCTP packet.  The total size of the resultant IP datagram,
including the SCTP packet and IP headers, MUST be less than or equal
to the current Path MTU.

---------
Old text: (Section 10.1)
---------
o  Receive Unacknowledged Message

   Format: RECEIVE_UNACKED(data retrieval id, buffer address, buffer
           size, [,stream id] [, stream sequence number] [,partial
           flag] [,payload protocol-id])

---------
New text: (Section 10.1)
---------

O) Receive Unacknowledged Message

   Format: RECEIVE_UNACKED(data retrieval id, buffer address, buffer
           size, [,stream id] [, stream sequence number] [,partial
           flag] [,payload protocol-id])
```

```
---------
Old text: (Appendix C)
---------
ICMP2) An implementation MAY ignore all ICMPv6 messages where the
       type field is not "Destination Unreachable", "Parameter
       Problem",, or "Packet Too Big".

---------
New text: (Appendix C)
---------

ICMP2) An implementation MAY ignore all ICMPv6 messages where the
       type field is not "Destination Unreachable", "Parameter
       Problem", or "Packet Too Big".
```

3.9.3. Solution Description

Typos fixed.

3.10. CRC32c Sample Code

3.10.1. Description of the Problem

The CRC32c computation is described in Appendix B of [RFC4960]. However, the corresponding sample code and its explanation appears at the end of Appendix C, which deals with ICMP handling.

3.10.2. Text Changes to the Document

Move the sample code related to CRC32c computation and its explanation from the end of Appendix C to the end of Appendix B.

3.10.3. Solution Description

Text moved to the appropriate location.

3.11. partial_bytes_acked after T3-rtx Expiration

3.11.1. Description of the Problem

Section 7.2.3 of [RFC4960] explicitly states that partial_bytes_acked should be reset to 0 after packet loss detecting from SACK but the same is missed for T3-rtx timer expiration.

3.11.2. Text Changes to the Document

```
---------
Old text: (Section 7.2.3)
---------

When the T3-rtx timer expires on an address, SCTP should perform slow
start by:

ssthresh = max(cwnd/2, 4*MTU)
cwnd = 1*MTU

---------
New text: (Section 7.2.3)
---------

When the T3-rtx timer expires on an address, SCTP should perform slow
start by:

ssthresh = max(cwnd/2, 4*MTU)
cwnd = 1*MTU
partial_bytes_acked = 0
```

3.11.3. Solution Description

Specify that partial_bytes_acked should be reset to 0 after T3-rtx timer expiration.

3.12. Order of Adjustments of partial_bytes_acked and cwnd

3.12.1. Description of the Problem

Section 7.2.2 of [RFC4960] is unclear about the order of adjustments applied to partial_bytes_acked and cwnd in the congestion avoidance phase.

3.12.2. Text Changes to the Document

```
---------
Old text: (Section 7.2.2)
---------

o  When partial_bytes_acked is equal to or greater than cwnd and
   before the arrival of the SACK the sender had cwnd or more bytes
   of data outstanding (i.e., before arrival of the SACK, flightsize
   was greater than or equal to cwnd), increase cwnd by MTU, and
   reset partial_bytes_acked to (partial_bytes_acked - cwnd).

---------
New text: (Section 7.2.2)
---------

o  When partial_bytes_acked is equal to or greater than cwnd and
   before the arrival of the SACK the sender had cwnd or more bytes
   of data outstanding (i.e., before arrival of the SACK, flightsize
   was greater than or equal to cwnd), partial_bytes_acked is reset
   to (partial_bytes_acked - cwnd). Next, cwnd is increased by MTU.
```

3.12.3. Solution Description

The new text defines the exact order of adjustments of partial_bytes_acked and cwnd in the congestion avoidance phase.

3.13. HEARTBEAT ACK and the association error counter

3.13.1. Description of the Problem

Section 8.1 and Section 8.3 of [RFC4960] prescribe that the receiver of a HEARTBEAT ACK must reset the association overall error counter. In some circumstances, e.g. when a router discards DATA chunks but not HEARTBEAT chunks due to the larger size of the DATA chunk, it might be better to not clear the association error counter on reception of the HEARTBEAT ACK and reset it only on reception of the SACK to avoid stalling the association.

3.13.2. Text Changes to the Document

Old text: (Section 8.1)

The counter shall be reset each time a DATA chunk sent to that peer endpoint is acknowledged (by the reception of a SACK) or a HEARTBEAT ACK is received from the peer endpoint.

New text: (Section 8.1)

The counter shall be reset each time a DATA chunk sent to that peer endpoint is acknowledged (by the reception of a SACK). When a HEARTBEAT ACK is received from the peer endpoint, the counter should also be reset. The receiver of the HEARTBEAT ACK may choose not to clear the counter if there is outstanding data on the association. This allows for handling the possible difference in reachability based on DATA chunks and HEARTBEAT chunks.

Old text: (Section 8.3)

Upon the receipt of the HEARTBEAT ACK, the sender of the HEARTBEAT should clear the error counter of the destination transport address to which the HEARTBEAT was sent, and mark the destination transport address as active if it is not so marked. The endpoint may optionally report to the upper layer when an inactive destination address is marked as active due to the reception of the latest HEARTBEAT ACK. The receiver of the HEARTBEAT ACK must also clear the association overall error count as well (as defined in Section 8.1).

New text: (Section 8.3)

Upon the receipt of the HEARTBEAT ACK, the sender of the HEARTBEAT should clear the error counter of the destination transport address to which the HEARTBEAT was sent, and mark the destination transport address as active if it is not so marked. The endpoint may optionally report to the upper layer when an inactive destination address is marked as active due to the reception of the latest HEARTBEAT ACK. The receiver of the HEARTBEAT ACK should also clear the association overall error counter (as defined in Section 8.1).

3.13.3. Solution Description

The new text provides a possibility to not reset the association overall error counter when a HEARTBEAT ACK is received if there are valid reasons for it.

3.14. Path for Fast Retransmission

3.14.1. Description of the Problem

[RFC4960] clearly describes where to retransmit data that is timed out when the peer is multi-homed but the same is not stated for fast retransmissions.

3.14.2. Text Changes to the Document

```
---------
Old text: (Section 6.4)
---------
```

Furthermore, when its peer is multi-homed, an endpoint SHOULD try to retransmit a chunk that timed out to an active destination transport address that is different from the last destination address to which the DATA chunk was sent.

```
---------
New text: (Section 6.4)
---------
```

Furthermore, when its peer is multi-homed, an endpoint SHOULD try to retransmit a chunk that timed out to an active destination transport address that is different from the last destination address to which the DATA chunk was sent.

When its peer is multi-homed, an endpoint SHOULD send fast retransmissions to the same destination transport address where original data was sent to. If the primary path has been changed and original data was sent there before the fast retransmit, the implementation MAY send it to the new primary path.

3.14.3. Solution Description

The new text clarifies where to send fast retransmissions.

3.15. Transmittal in Fast Recovery

3.15.1. Description of the Problem

The Fast Retransmit on Gap Reports algorithm intends that only the very first packet may be sent regardless of cwnd in the Fast Recovery phase but rule 3) of [RFC4960], Section 7.2.4, misses this clarification.

3.15.2. Text Changes to the Document

```
---------
Old text: (Section 7.2.4)
---------

3)  Determine how many of the earliest (i.e., lowest TSN) DATA chunks
    marked for retransmission will fit into a single packet, subject
    to constraint of the path MTU of the destination transport
    address to which the packet is being sent.  Call this value K.
    Retransmit those K DATA chunks in a single packet.  When a Fast
    Retransmit is being performed, the sender SHOULD ignore the value
    of cwnd and SHOULD NOT delay retransmission for this single
    packet.

---------
New text: (Section 7.2.4)
---------

3)  If not in Fast Recovery, determine how many of the earliest
    (i.e., lowest TSN) DATA chunks marked for retransmission will fit
    into a single packet, subject to constraint of the path MTU of
    the destination transport address to which the packet is being
    sent. Call this value K. Retransmit those K DATA chunks in a
    single packet. When a Fast Retransmit is being performed, the
    sender SHOULD ignore the value of cwnd and SHOULD NOT delay
    retransmission for this single packet.
```

3.15.3. Solution Description

The new text explicitly specifies to send only the first packet in the Fast Recovery phase disregarding cwnd limitations.

3.16. Initial Value of ssthresh

3.16.1. Description of the Problem

The initial value of ssthresh should be set arbitrarily high. Using the advertised receiver window of the peer is inappropriate if the peer increases its window after the handshake. Furthermore, use a higher requirements level, since not following the advice may result in performance problems.

3.16.2. Text Changes to the Document

```
---------
Old text: (Section 7.2.1)
---------

o  The initial value of ssthresh MAY be arbitrarily high (for
   example, implementations MAY use the size of the receiver
   advertised window).

---------
New text: (Section 7.2.1)
---------

o  The initial value of ssthresh SHOULD be arbitrarily high (e.g.,
   to the size of the largest possible advertised window).
```

3.16.3. Solution Description

Use the same value as suggested in [RFC5681], Section 3.1, as an appropriate initial value. Furthermore use the same requirements level.

3.17. Automatically Confirmed Addresses

3.17.1. Description of the Problem

The Path Verification procedure of [RFC4960] prescribes that any address passed to the sender of the INIT by its upper layer is automatically CONFIRMED. This however is unclear if only addresses in the request to initiate association establishment are considered or any addresses provided by the upper layer in any requests (e.g. in 'Set Primary').

3.17.2. Text Changes to the Document

```
---------
Old text: (Section 5.4)
---------

1)  Any address passed to the sender of the INIT by its upper layer
   is automatically considered to be CONFIRMED.

---------
New text: (Section 5.4)
---------

1)  Any addresses passed to the sender of the INIT by its upper
   layer in the request to initialize an association is
   automatically considered to be CONFIRMED.
```

3.17.3. Solution Description

The new text clarifies that only addresses provided by the upper layer in the request to initialize an association are automatically confirmed.

3.18. Only One Packet after Retransmission Timeout

3.18.1. Description of the Problem

[RFC4960] is not completely clear when it describes data transmission after T3-rtx timer expiration. Section 7.2.1 does not specify how many packets are allowed to be sent after T3-rtx timer expiration if more than one packet fit into cwnd. At the same time, Section 7.2.3 has the text without normative language saying that SCTP should ensure that no more than one packet will be in flight after T3-rtx timer expiration until successful acknowledgment. It makes the text inconsistent.

3.18.2. Text Changes to the Document

```
---------
Old text: (Section 7.2.1)
---------

o  The initial cwnd after a retransmission timeout MUST be no more
   than 1*MTU.

---------
New text: (Section 7.2.1)
---------

o  The initial cwnd after a retransmission timeout MUST be no more
   than 1*MTU and only one packet is allowed to be in flight
   until successful acknowledgement.
```

3.18.3. Solution Description

The new text clearly specifies that only one packet is allowed to be sent after T3-rtx timer expiration until successful acknowledgement.

3.19. INIT ACK Path for INIT in COOKIE-WAIT State

3.19.1. Description of the Problem

In case of an INIT received in the COOKIE-WAIT state [RFC4960] prescribes to send an INIT ACK to the same destination address to which the original INIT has been sent. This text does not address the possibility of the upper layer to provide multiple remote IP addresses while requesting the association establishment. If the upper layer has provided multiple IP addresses and only a subset of these addresses are supported by the peer then the destination address of the original INIT may be absent in the incoming INIT and sending INIT ACK to that address is useless.

3.19.2. Text Changes to the Document

```
---------
Old text: (Section 5.2.1)
---------
```

Upon receipt of an INIT in the COOKIE-WAIT state, an endpoint MUST respond with an INIT ACK using the same parameters it sent in its original INIT chunk (including its Initiate Tag, unchanged). When responding, the endpoint MUST send the INIT ACK back to the same address that the original INIT (sent by this endpoint) was sent.

```
---------
New text: (Section 5.2.1)
---------
```

Upon receipt of an INIT in the COOKIE-WAIT state, an endpoint MUST respond with an INIT ACK using the same parameters it sent in its original INIT chunk (including its Initiate Tag, unchanged). When responding, the following rules MUST be applied:

1) The INIT ACK MUST only be sent to an address passed by the upper layer in the request to initialize the association.

2) The INIT ACK MUST only be sent to an address reported in the incoming INIT.

3) The INIT ACK SHOULD be sent to the source address of the received INIT.

3.19.3. Solution Description

The new text requires sending INIT ACK to the destination address that is passed by the upper layer and reported in the incoming INIT. If the source address of the INIT fulfills it then sending the INIT ACK to the source address of the INIT is the preferred behavior.

3.20. Zero Window Probing and Unreachable Primary Path

3.20.1. Description of the Problem

Section 6.1 of [RFC4960] states that when sending zero window probes, SCTP should neither increment the association counter nor increment the destination address error counter if it continues to receive new packets from the peer. But receiving new packets from the peer does not guarantee peer’s accessibility and, if the destination address becomes unreachable during zero window probing, SCTP cannot get a changed rwnd until it switches the destination address for probes.

3.20.2. Text Changes to the Document

```
---------
Old text: (Section 6.1)
---------

If the sender continues to receive new packets from the receiver
while doing zero window probing, the unacknowledged window probes
should not increment the error counter for the association or any
destination transport address.  This is because the receiver MAY
keep its window closed for an indefinite time.  Refer to Section
6.2 on the receiver behavior when it advertises a zero window.

---------
New text: (Section 6.1)
---------

If the sender continues to receive SACKs from the peer
while doing zero window probing, the unacknowledged window probes
should not increment the error counter for the association or any
destination transport address.  This is because the receiver MAY
keep its window closed for an indefinite time.  Refer to Section
6.2 on the receiver behavior when it advertises a zero window.
```

3.20.3. Solution Description

The new text clarifies that if the receiver continues to send SACKs, the sender of probes should not increment the error counter of the association and the destination address even if the SACKs do not acknowledge the probes.

3.21. Normative Language in Section 10

3.21.1. Description of the Problem

Section 10 of [RFC4960] is informative and normative language such as MUST and MAY cannot be used there. However, there are several places in Section 10 where MUST and MAY are used.

3.21.2. Text Changes to the Document

```
---------
Old text: (Section 10.1)
---------

E) Send

 Format: SEND(association id, buffer address, byte count [,context]
```

```
            [,stream id] [,life time] [,destination transport address]
            [,unordered flag] [,no-bundle flag] [,payload protocol-id] )
 -> result

...

o  no-bundle flag - instructs SCTP not to bundle this user data with
   other outbound DATA chunks.  SCTP MAY still bundle even when this
   flag is present, when faced with network congestion.

---------
New text: (Section 10.1)
---------

E) Send

 Format: SEND(association id, buffer address, byte count [,context]
            [,stream id] [,life time] [,destination transport address]
            [,unordered flag] [,no-bundle flag] [,payload protocol-id] )
 -> result

...

o  no-bundle flag - instructs SCTP not to bundle this user data with
   other outbound DATA chunks.  SCTP may still bundle even when this
   flag is present, when faced with network congestion.

---------
Old text: (Section 10.1)
---------

G) Receive

 Format: RECEIVE(association id, buffer address, buffer size
            [,stream id])
 -> byte count [,transport address] [,stream id] [,stream sequence
    number] [,partial flag] [,delivery number] [,payload protocol-id]

...

o  partial flag - if this returned flag is set to 1, then this
   Receive contains a partial delivery of the whole message.  When
   this flag is set, the stream id and Stream Sequence Number MUST
   accompany this receive.  When this flag is set to 0, it indicates
   that no more deliveries will be received for this Stream Sequence
   Number.

---------
```

New text: (Section 10.1)

G) Receive

```
 Format: RECEIVE(association id, buffer address, buffer size
         [,stream id])
 -> byte count [,transport address] [,stream id] [,stream sequence
    number] [,partial flag] [,delivery number] [,payload protocol-id]
```

...

o partial flag - if this returned flag is set to 1, then this Receive contains a partial delivery of the whole message. When this flag is set, the stream id and Stream Sequence Number must accompany this receive. When this flag is set to 0, it indicates that no more deliveries will be received for this Stream Sequence Number.

Old text: (Section 10.1)

N) Receive Unsent Message

```
   Format: RECEIVE_UNSENT(data retrieval id, buffer address, buffer
           size [,stream id] [, stream sequence number] [,partial
           flag] [,payload protocol-id])
```

...

o partial flag - if this returned flag is set to 1, then this message is a partial delivery of the whole message. When this flag is set, the stream id and Stream Sequence Number MUST accompany this receive. When this flag is set to 0, it indicates that no more deliveries will be received for this Stream Sequence Number.

New text: (Section 10.1)

N) Receive Unsent Message

```
   Format: RECEIVE_UNSENT(data retrieval id, buffer address, buffer
           size [,stream id] [, stream sequence number] [,partial
           flag] [,payload protocol-id])
```

```
...

o  partial flag - if this returned flag is set to 1, then this
   message is a partial delivery of the whole message.  When this
   flag is set, the stream id and Stream Sequence Number must
   accompany this receive.  When this flag is set to 0, it indicates
   that no more deliveries will be received for this Stream Sequence
   Number.

---------
Old text: (Section 10.1)
---------

O) Receive Unacknowledged Message

   Format: RECEIVE_UNACKED(data retrieval id, buffer address, buffer
           size, [,stream id] [, stream sequence number] [,partial
           flag] [,payload protocol-id])

...

o  partial flag - if this returned flag is set to 1, then this
   message is a partial delivery of the whole message.  When this
   flag is set, the stream id and Stream Sequence Number MUST
   accompany this receive.  When this flag is set to 0, it indicates
   that no more deliveries will be received for this Stream Sequence
   Number.

---------
New text: (Section 10.1)
---------

O) Receive Unacknowledged Message

   Format: RECEIVE_UNACKED(data retrieval id, buffer address, buffer
           size, [,stream id] [, stream sequence number] [,partial
           flag] [,payload protocol-id])

...

o  partial flag - if this returned flag is set to 1, then this
   message is a partial delivery of the whole message.  When this
   flag is set, the stream id and Stream Sequence Number must
   accompany this receive.  When this flag is set to 0, it indicates
   that no more deliveries will be received for this Stream Sequence
   Number.
```

3.21.3. Solution Description

The normative language is removed from Section 10.

3.22. Increase of partial_bytes_acked in Congestion Avoidance

3.22.1. Description of the Problem

Two issues have been discovered with the partial_bytes_acked handling described in Section 7.2.2 of [RFC4960]:

- o If the Cumulative TSN Ack Point is not advanced but the SACK chunk acknowledges new TSNs in the Gap Ack Blocks, these newly acknowledged TSNs are not considered for partial_bytes_acked although these TSNs were successfully received by the peer.
- o Duplicate TSNs are not considered in partial_bytes_acked although they confirm that the DATA chunks were successfully received by the peer.

3.22.2. Text Changes to the Document

```
---------
Old text: (Section 7.2.2)
---------

o  Whenever cwnd is greater than ssthresh, upon each SACK arrival
   that advances the Cumulative TSN Ack Point, increase
   partial_bytes_acked by the total number of bytes of all new chunks
   acknowledged in that SACK including chunks acknowledged by the new
   Cumulative TSN Ack and by Gap Ack Blocks.

---------
New text: (Section 7.2.2)
---------

o  Whenever cwnd is greater than ssthresh, upon each SACK arrival,
   increase partial_bytes_acked by the total number of bytes of all
   new chunks acknowledged in that SACK including chunks acknowledged
   by the new Cumulative TSN Ack, by Gap Ack Blocks and by the number
   of bytes of duplicated chunks reported in Duplicate TSNs.
```

3.22.3. Solution Description

Now partial_bytes_acked is increased by TSNs reported as duplicated as well as TSNs newly acknowledged in Gap Ack Blocks even if the Cumulative TSN Ack Point is not advanced.

3.23. Inconsistency in Notifications Handling

3.23.1. Description of the Problem

[RFC4960] uses inconsistent normative and non-normative language when describing rules for sending notifications to the upper layer. E.g. Section 8.2 of [RFC4960] says that when a destination address becomes inactive due to an unacknowledged DATA chunk or HEARTBEAT chunk, SCTP SHOULD send a notification to the upper layer while Section 8.3 of [RFC4960] says that when a destination address becomes inactive due to an unacknowledged HEARTBEAT chunk, SCTP may send a notification to the upper layer.

This makes the text inconsistent.

3.23.2. Text Changes to the Document

The following cahnge is based on the change described in Section 3.6.

Old text: (Section 8.1)

An endpoint shall keep a counter on the total number of consecutive retransmissions to its peer (this includes data retransmissions to all the destination transport addresses of the peer if it is multi-homed), including the number of unacknowledged HEARTBEAT chunks observed on the path which currently is used for data transfer. Unacknowledged HEARTBEAT chunks observed on paths different from the path currently used for data transfer shall not increment the association error counter, as this could lead to association closure even if the path which currently is used for data transfer is available (but idle). If the value of this counter exceeds the limit indicated in the protocol parameter 'Association.Max.Retrans', the endpoint shall consider the peer endpoint unreachable and shall stop transmitting any more data to it (and thus the association enters the CLOSED state). In addition, the endpoint MAY report the failure to the upper layer and optionally report back all outstanding user data remaining in its outbound queue. The association is automatically closed when the peer endpoint becomes unreachable.

New text: (Section 8.1)

An endpoint shall keep a counter on the total number of consecutive retransmissions to its peer (this includes data retransmissions to all the destination transport addresses of the peer if it is multi-homed), including the number of unacknowledged HEARTBEAT chunks observed on the path which currently is used for data transfer. Unacknowledged HEARTBEAT chunks observed on paths different from the path currently used for data transfer shall not increment the association error counter, as this could lead to association closure even if the path which currently is used for data transfer is available (but idle). If the value of this counter exceeds the limit indicated in the protocol parameter 'Association.Max.Retrans', the endpoint shall consider the peer endpoint unreachable and shall stop transmitting any more data to it (and thus the association enters the CLOSED state). In addition, the endpoint SHOULD report the failure to the upper layer and optionally report back all outstanding user data remaining in its outbound queue. The association is automatically closed when the peer endpoint becomes unreachable.

The following changes are based on [RFC4960].

Old text: (Section 8.2)

When an outstanding TSN is acknowledged or a HEARTBEAT sent to that address is acknowledged with a HEARTBEAT ACK, the endpoint shall clear the error counter of the destination transport address to which the DATA chunk was last sent (or HEARTBEAT was sent). When the peer endpoint is multi-homed and the last chunk sent to it was a retransmission to an alternate address, there exists an ambiguity as to whether or not the acknowledgement should be credited to the address of the last chunk sent. However, this ambiguity does not seem to bear any significant consequence to SCTP behavior. If this ambiguity is undesirable, the transmitter may choose not to clear the error counter if the last chunk sent was a retransmission.

New text: (Section 8.2)

When an outstanding TSN is acknowledged or a HEARTBEAT sent to that address is acknowledged with a HEARTBEAT ACK, the endpoint shall clear the error counter of the destination transport address to which the DATA chunk was last sent (or HEARTBEAT was sent), and SHOULD also report to the upper layer when an inactive destination address is marked as active. When the peer endpoint is multi-homed and the last chunk sent to it was a retransmission to an alternate address, there exists an ambiguity as to whether or not the acknowledgement should be credited to the address of the last chunk sent. However, this ambiguity does not seem to bear any significant consequence to SCTP behavior. If this ambiguity is undesirable, the transmitter may choose not to clear the error counter if the last chunk sent was a retransmission.

Old text: (Section 8.3)

When the value of this counter reaches the protocol parameter 'Path.Max.Retrans', the endpoint should mark the corresponding destination address as inactive if it is not so marked, and may also optionally report to the upper layer the change of reachability of this destination address. After this, the endpoint should continue HEARTBEAT on this destination address but should stop increasing the counter.

New text: (Section 8.3)

When the value of this counter exceeds the protocol parameter 'Path.Max.Retrans', the endpoint should mark the corresponding destination address as inactive if it is not so marked, and SHOULD also report to the upper layer the change of reachability of this destination address. After this, the endpoint should continue HEARTBEAT on this destination address but should stop increasing the counter.

Old text: (Section 8.3)

Upon the receipt of the HEARTBEAT ACK, the sender of the HEARTBEAT should clear the error counter of the destination transport address to which the HEARTBEAT was sent, and mark the destination transport address as active if it is not so marked. The endpoint may optionally report to the upper layer when an inactive destination address is marked as active due to the reception of the latest HEARTBEAT ACK. The receiver of the HEARTBEAT ACK must also clear the association overall error count as well (as defined in Section 8.1).

New text: (Section 8.3)

Upon the receipt of the HEARTBEAT ACK, the sender of the HEARTBEAT should clear the error counter of the destination transport address to which the HEARTBEAT was sent, and mark the destination transport address as active if it is not so marked. The endpoint SHOULD report to the upper layer when an inactive destination address is marked as active due to the reception of the latest HEARTBEAT ACK. The receiver of the HEARTBEAT ACK should also clear the association overall error counter (as defined in Section 8.1).

Old text: (Section 9.2)

An endpoint should limit the number of retransmissions of the SHUTDOWN chunk to the protocol parameter 'Association.Max.Retrans'. If this threshold is exceeded, the endpoint should destroy the TCB and MUST report the peer endpoint unreachable to the upper layer (and thus the association enters the CLOSED state).

New text: (Section 9.2)

An endpoint should limit the number of retransmissions of the SHUTDOWN chunk to the protocol parameter 'Association.Max.Retrans'. If this threshold is exceeded, the endpoint should destroy the TCB and SHOULD report the peer endpoint unreachable to the upper layer (and thus the association enters the CLOSED state).

Old text: (Section 9.2)

The sender of the SHUTDOWN ACK should limit the number of retransmissions of the SHUTDOWN ACK chunk to the protocol parameter 'Association.Max.Retrans'. If this threshold is exceeded, the endpoint should destroy the TCB and may report the peer endpoint unreachable to the upper layer (and thus the association enters the CLOSED state).

New text: (Section 9.2)

The sender of the SHUTDOWN ACK should limit the number of retransmissions of the SHUTDOWN ACK chunk to the protocol parameter 'Association.Max.Retrans'. If this threshold is exceeded, the endpoint should destroy the TCB and SHOULD report the peer endpoint unreachable to the upper layer (and thus the association enters the CLOSED state).

3.23.3. Solution Description

The inconsistencies are removed by using consistently SHOULD.

3.24. SACK.Delay Not Listed as a Protocol Parameter

3.24.1. Description of the Problem

SCTP as specified in [RFC4960] supports delaying SACKs. The timer value for this is a parameter and Section 6.2 of [RFC4960] specifies a default and maximum value for it. However, defining a name for this parameter and listing it in the table of protocol parameters in Section 15 of [RFC4960] is missing.

This issue was reported as an Errata for [RFC4960] with Errata ID 4656.

3.24.2. Text Changes to the Document

```
---------
Old text: (Section 6.2)
---------

An implementation MUST NOT allow the maximum delay to be configured
to be more than 500 ms.  In other words, an implementation MAY lower
this value below 500 ms but MUST NOT raise it above 500 ms.

---------
New text: (Section 6.2)
---------

An implementation MUST NOT allow the maximum delay (protocol
parameter 'SACK.Delay') to be configured to be more than 500 ms.
In other words, an implementation MAY lower the value of
SACK.Delay below 500 ms but MUST NOT raise it above 500 ms.

---------
Old text: (Section 15)
---------

The following protocol parameters are RECOMMENDED:

   RTO.Initial - 3 seconds
   RTO.Min - 1 second
   RTO.Max - 60 seconds
   Max.Burst - 4
   RTO.Alpha - 1/8
   RTO.Beta - 1/4
   Valid.Cookie.Life - 60 seconds
   Association.Max.Retrans - 10 attempts
   Path.Max.Retrans - 5 attempts (per destination address)
   Max.Init.Retransmits - 8 attempts
   HB.interval - 30 seconds
   HB.Max.Burst - 1

---------
New text: (Section 15)
---------

The following protocol parameters are RECOMMENDED:

   RTO.Initial - 3 seconds
   RTO.Min - 1 second
   RTO.Max - 60 seconds
   Max.Burst - 4
```

```
RTO.Alpha - 1/8
RTO.Beta - 1/4
Valid.Cookie.Life - 60 seconds
Association.Max.Retrans - 10 attempts
Path.Max.Retrans - 5 attempts (per destination address)
Max.Init.Retransmits - 8 attempts
HB.interval - 30 seconds
HB.Max.Burst - 1
SACK.Delay - 200 milliseconds
```

3.24.3. Solution Description

The parameter was given a name and added to the list of protocol parameters.

3.25. Processing of Chunks in an Incoming SCTP Packet

3.25.1. Description of the Problem

There are a few places in [RFC4960] where the receiver of a packet must discard it while processing the chunks of the packet. It is unclear whether the receiver has to rollback state changes already performed while processing the packet or not.

The intention of [RFC4960] is to process an incoming packet chunk by chunk and do not perform any prescreening of chunks in the received packet so the receiver must only discard a chunk causing discard and all further chunks.

3.25.2. Text Changes to the Document

```
---------
Old text: (Section 3.2)
---------

00 -  Stop processing this SCTP packet and discard it, do not
      process any further chunks within it.

01 -  Stop processing this SCTP packet and discard it, do not
      process any further chunks within it, and report the
      unrecognized chunk in an 'Unrecognized Chunk Type'.

---------
New text: (Section 3.2)
---------

00 -  Stop processing this SCTP packet, discard the unrecognized
      chunk and all further chunks.
```

01 - Stop processing this SCTP packet, discard the unrecognized chunk and all further chunks, and report the unrecognized chunk in an 'Unrecognized Chunk Type'.

Old text: (Section 11.3)

It is helpful for some firewalls if they can inspect just the first fragment of a fragmented SCTP packet and unambiguously determine whether it corresponds to an INIT chunk (for further information, please refer to [RFC1858]). Accordingly, we stress the requirements, stated in Section 3.1, that (1) an INIT chunk MUST NOT be bundled with any other chunk in a packet, and (2) a packet containing an INIT chunk MUST have a zero Verification Tag. Furthermore, we require that the receiver of an INIT chunk MUST enforce these rules by silently discarding an arriving packet with an INIT chunk that is bundled with other chunks or has a non-zero verification tag and contains an INIT-chunk.

New text: (Section 11.3)

It is helpful for some firewalls if they can inspect just the first fragment of a fragmented SCTP packet and unambiguously determine whether it corresponds to an INIT chunk (for further information, please refer to [RFC1858]). Accordingly, we stress the requirements, stated in Section 3.1, that (1) an INIT chunk MUST NOT be bundled with any other chunk in a packet, and (2) a packet containing an INIT chunk MUST have a zero Verification Tag. Furthermore, we require that the receiver of an INIT chunk MUST enforce these rules by silently discarding the INIT chunk and all further chunks if the INIT chunk is bundled with other chunks or the packet has a non-zero verification tag.

3.25.3. Solution Description

The new text makes it clear that chunks can be processed from the beginning to the end and no rollback or pre-screening is required.

3.26. CWND Increase in Congestion Avoidance Phase

3.26.1. Description of the Problem

[RFC4960] in Section 7.2.2 prescribes to increase cwnd by 1*MTU per RTT if the sender has cwnd or more bytes of outstanding data to the corresponding address in the Congestion Avoidance phase. However,

this is described without normative language. Moreover, Section 7.2.2 includes an algorithm how an implementation can achieve it but this algorithm is underspecified and actually allows increasing cwnd by more than 1*MTU per RTT.

3.26.2. Text Changes to the Document

Old text: (Section 7.2.2)

When cwnd is greater than ssthresh, cwnd should be incremented by 1*MTU per RTT if the sender has cwnd or more bytes of data outstanding for the corresponding transport address.

New text: (Section 7.2.2)

When cwnd is greater than ssthresh, cwnd should be incremented by 1*MTU per RTT if the sender has cwnd or more bytes of data outstanding for the corresponding transport address. The basic guidelines for incrementing cwnd during congestion avoidance are:

- o SCTP MAY increment cwnd by 1*MTU.

- o SCTP SHOULD increment cwnd by one 1*MTU once per RTT when the sender has cwnd or more bytes of data outstanding for the corresponding transport address.

- o SCTP MUST NOT increment cwnd by more than 1*MTU per RTT.

Old text: (Section 7.2.2)

- o Whenever cwnd is greater than ssthresh, upon each SACK arrival that advances the Cumulative TSN Ack Point, increase partial_bytes_acked by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack and by Gap Ack Blocks.

- o When partial_bytes_acked is equal to or greater than cwnd and before the arrival of the SACK the sender had cwnd or more bytes of data outstanding (i.e., before arrival of the SACK, flightsize was greater than or equal to cwnd), increase cwnd by MTU, and reset partial_bytes_acked to (partial_bytes_acked - cwnd).

```
---------
New text: (Section 7.2.2)
---------
```

- o Whenever cwnd is greater than ssthresh, upon each SACK arrival, increase partial_bytes_acked by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack, by Gap Ack Blocks and by the number of bytes of duplicated chunks reported in Duplicate TSNs.

- o When partial_bytes_acked is greater than cwnd and before the arrival of the SACK the sender had less bytes of data outstanding than cwnd (i.e., before arrival of the SACK, flightsize was less than cwnd), reset partial_bytes_acked to cwnd.

- o When partial_bytes_acked is equal to or greater than cwnd and before the arrival of the SACK the sender had cwnd or more bytes of data outstanding (i.e., before arrival of the SACK, flightsize was greater than or equal to cwnd), partial_bytes_acked is reset to (partial_bytes_acked - cwnd). Next, cwnd is increased by MTU.

3.26.3. Solution Description

The basic guidelines for incrementing cwnd during congestion avoidance phase are added into Section 7.2.2. The guidelines include the normative language and are aligned with [RFC5681].

The algorithm from Section 7.2.2 is improved to not allow increasing cwnd by more than 1*MTU per RTT.

3.27. Refresh of cwnd and ssthresh after Idle Period

3.27.1. Description of the Problem

[RFC4960] prescribes to adjust cwnd per RTO if the endpoint does not transmit data on a given transport address. In addition to that, it prescribes to set cwnd to the initial value after a sufficiently long idle period. The latter is excessive. Moreover, it is unclear what is a sufficiently long idle period.

[RFC4960] doesn’t specify the handling of ssthresh in the idle case. If ssthres is reduced due to a packet loss, ssthresh is never recovered. So traffic can end up in Congestion Avoidance all the time, resulting in a low sending rate and bad performance. The problem is even more serious for SCTP because in a multi-homed SCTP association traffic switch back to the previously failed primary path will also lead to the situation where traffic ends up in Congestion Avoidance.

3.27.2. Text Changes to the Document

```
---------
Old text: (Section 7.2.1)
---------

o  The initial cwnd before DATA transmission or after a sufficiently
   long idle period MUST be set to min(4*MTU, max (2*MTU, 4380
   bytes)).

---------
New text: (Section 7.2.1)
---------

o  The initial cwnd before DATA transmission MUST be set to
   min(4*MTU, max (2*MTU, 4380 bytes)).

---------
Old text: (Section 7.2.1)
---------

o  When the endpoint does not transmit data on a given transport
   address, the cwnd of the transport address should be adjusted to
   max(cwnd/2, 4*MTU) per RTO.

---------
New text: (Section 7.2.1)
---------
o  When the endpoint does not transmit data on a given transport
   address, the cwnd of the transport address should be adjusted to
   max(cwnd/2, 4*MTU) per RTO. At the first cwnd adjustment, the
   ssthresh of the transport address should be adjusted to the cwnd.
```

3.27.3. Solution Description

A rule about cwnd adjustment after a sufficiently long idle period is removed.

The text is updated to refresh ssthresh after the idle period. When the idle period is detected, the cwnd value is stored to the ssthresh value.

3.28. Window Updates After Receiver Window Opens Up

3.28.1. Description of the Problem

The sending of SACK chunks for window updates is only indirectly referenced in [RFC4960], Section 6.2, where it is stated that an SCTP receiver must not generate more than one SACK for every incoming packet, other than to update the offered window.

However, the sending of window updates when the receiver window opens up is necessary to avoid performance problems.

3.28.2. Text Changes to the Document

```
---------
Old text: (Section 6.2)
---------
```

An SCTP receiver MUST NOT generate more than one SACK for every incoming packet, other than to update the offered window as the receiving application consumes new data.

```
---------
New text: (Section 6.2)
---------
```

An SCTP receiver MUST NOT generate more than one SACK for every incoming packet, other than to update the offered window as the receiving application consumes new data. When the window opens up, an SCTP receiver SHOULD send additional SACK chunks to update the window even if no new data is received. The receiver MUST avoid sending large burst of window updates.

3.28.3. Solution Description

The new text makes clear that additional SACK chunks for window updates may be sent as long as excessive bursts are avoided.

3.29. Path of DATA and Reply Chunks

3.29.1. Description of the Problem

Section 6.4 of [RFC4960] describes the transmission policy for multi-homed SCTP endpoints. However, there are the following issues with it:

- o It states that a SACK should be sent to the source address of an incoming DATA. However, it is known that other SACK policies

(e.g. sending SACKs always to the primary path) may be more beneficial in some situations.

- o Initially it states that an endpoint should always transmit DATA chunks to the primary path. Then it states that the rule for transmittal of reply chunks should also be followed if the endpoint is bundling DATA chunks together with the reply chunk which contradicts with the first statement to always transmit DATA chunks to the primary path. Some implementations were having problems with it and sent DATA chunks bundled with reply chunks to a different destination address than the primary path that caused many gaps.

3.29.2. Text Changes to the Document

Old text: (Section 6.4)

An endpoint SHOULD transmit reply chunks (e.g., SACK, HEARTBEAT ACK, etc.) to the same destination transport address from which it received the DATA or control chunk to which it is replying. This rule should also be followed if the endpoint is bundling DATA chunks together with the reply chunk.

However, when acknowledging multiple DATA chunks received in packets from different source addresses in a single SACK, the SACK chunk may be transmitted to one of the destination transport addresses from which the DATA or control chunks being acknowledged were received.

New text: (Section 6.4)

An endpoint SHOULD transmit reply chunks (e.g., INIT ACK, COOKIE ACK, HEARTBEAT ACK, etc.) in response to control chunks to the same destination transport address from which it received the control chunk to which it is replying.

The selection of the destination transport address for packets containing SACK chunks is implementation dependent. However, an endpoint SHOULD NOT vary the destination transport address of a SACK when it receives DATA chunks from the same source address.

When acknowledging multiple DATA chunks received in packets from different source addresses in a single SACK, the SACK chunk MAY be transmitted to one of the destination transport addresses from which the DATA or control chunks being acknowledged were received.

3.29.3. Solution Description

The SACK transmission policy is left implementation dependent but it is specified to not vary the destination address of a packet containing a SACK chunk unless there are reasons for it as it may negatively impact RTT measurement.

A confusing statement that prescribes to follow the rule for transmittal of reply chunks when the endpoint is bundling DATA chunks together with the reply chunk is removed.

4. IANA Considerations

This document does not require any actions from IANA.

5. Security Considerations

This document does not add any security considerations to those given in [RFC4960].

6. Acknowledgments

The authors wish to thank Pontus Andersson, Eric W. Biederman, Cedric Bonnet, Lionel Morand, Jeff Morriss, Karen E. E. Nielsen, Tom Petch and Julien Pourtet for their invaluable comments.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

7.2. Informative References

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, DOI 10.17487/RFC2960, October 2000, <http://www.rfc-editor.org/info/rfc2960>.

[RFC4460] Stewart, R., Arias-Rodriguez, I., Poon, K., Caro, A., and M. Tuexen, "Stream Control Transmission Protocol (SCTP) Specification Errata and Issues", RFC 4460, DOI 10.17487/RFC4460, April 2006, <http://www.rfc-editor.org/info/rfc4460>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

Authors' Addresses

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
United States

Email: randall@lakerest.net

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Maksim Proshin
Ericsson
Kistavaegen 25
Stockholm 164 80
Sweden

Email: mproshin@tieto.mera.ru