Internet Engineering Task Force
Internet-Draft
Updates: 3662,4594 (if approved)
Intended status: Standards Track
Expires: February 2, 2017

R. Bless
Karlsruhe Institute of Technology (KIT)

August 1, 2016

A Lower Effort Per-Hop Behavior (LE PHB)
draft-bless-tsvwg-le-phb-01

Abstract

This document specifies properties and characteristics of a Lower Effort (LE) per-hop behavior (PHB). The primary objective of this LE PHB is to protect best-effort (BE) traffic (packets forwarded with the default PHB) from LE traffic in congestion situations, i.e., when resources become scarce, best-effort traffic has precedence over LE traffic and may preempt it. There are numerous uses for this PHB, e.g., for background traffic of low precedence, such as bulk data transfers with low priority in time, non time-critical backups, larger software updates, web search engines while gathering information from web servers and so on. This document recommends a standard DSCP value for the LE PHB.

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 2, 2017.

Table of Contents

1. Introduction

This document defines a Differentiated Services per-hop behavior RFC 2474 [RFC2474] called "Lower Effort" (LE) which is intended for traffic of sufficiently low urgency, in which all other traffic takes precedence over LE traffic in consumption of network link bandwidth. Low urgency traffic has got a low priority in time, which does not necessarily imply that it is generally of minor importance. From this viewpoint, it can be considered as a network equivalent to a background priority for processes in an operating system. There may or may not be memory (buffer) resources allocated for this type of traffic.

Some networks carry traffic for which delivery is considered optional; that is, packets of this type of traffic ought to consume network resources only when no other traffic is present. Alternatively, the effect of this type of traffic on all other network traffic is strictly limited. This is distinct from "best-effort" (BE) traffic since the network makes no commitment to deliver LE packets. In contrast, BE traffic receives an implied "good faith" commitment of at least some available network resources. This

document proposes a Lower Effort Differentiated Services per-hop behavior (LE PHB) for handling this "optional" traffic in a differentiated services node.

1.1. Applicability

A Lower Effort PHB is for sending extremely non-critical traffic across a Differentiated Services (DS) domain or DS region. There should be an expectation that packets of the LE PHB may be delayed or dropped when any other traffic is present. Use of the LE PHB might assist a network operator in moving certain kinds of traffic or users to off-peak times. Alternatively, or in addition, packets can be designated for the LE PHB when the goal is to protect all other packet traffic from competition with the LE aggregate while not completely banning LE traffic from the network. An LE PHB should not be used for a customer's "normal internet" traffic nor should packets be "downgraded" to the LE PHB used as a substitute for dropping packets that ought simply to be dropped as unauthorized. The LE PHB is expected to have applicability in networks that have at least some unused capacity at some times of day.

This is a PHB that allows networks to protect themselves from selected types of traffic rather than giving a selected traffic aggregate preferential treatment. Moreover, it may also exploit all unused resources from other PHBs.

There is no intrinsic reason to limit the applicability of the LE PHB to any particular application or type of traffic. It is intended as an additional tool for administrators in engineering networks. For instance, it can be used for filling up protection capacity of transmission links which is otherwise unused. Some network providers keep link utilization below 50% in order to being able carrying all traffic without loss in case of rerouting due to a link failure. LE marked traffic can utilize the normally unused capacity and will be preempted automatically in case of link failure when 100% of the link capacity is required for all other traffic. Ideally, applications mark their packets as LE traffic, since they know the urgency of flows.

Example uses for the LE PHB comprise:

- For traffic caused by world-wide web search engines while they gather information from web servers.

- For software updates or dissemination of new releases of operating systems.

- For backup traffic or non-time critical sychronization or mirroring traffic.
- For content distribution transfers between caches.
- For Netnews and other "bulk mail" of the Internet.
- For "downgraded" traffic from some other PHB when this does not violate the operational objectives of the other PHB or the overall network. LE should not be used for the general case of downgraded traffic, but may be used by design, e.g., to protect an internal network from untrusted external traffic sources. In this case there is no way for attackers to preempt internal (non LE) traffic by flooding. Another use case is mentioned in [RFC3754]: non-admitted multicast traffic.

1.2. Deployment Considerations

Internet-wide deployment of the LE PHB is eased by the following properties:

- No harm to other traffic: since the LE PHB has got the lowest priority it does not take resources from other PHBs. Deployment across different provider domains causes no trust issues or attack vectors to existing traffic.
- No parameters or configuration: the LE PHB requires no parameters and no configuration of traffic profiles and so on.
- No traffic conditioning mechanisms: the LE PHB requires only a queue and a scheduling mechanism, but no traffic meters, droppers or shapers.

Since LE traffic may be starved completely for a longer period of time, transport protocols or applications should be able to detect such a situation and should resume the transfer as soon as possible.

1.3. 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].

2. PHB Description

This PHB is defined in relation to the default PHB (best-effort). A packet forwarded with this PHB SHOULD have lower precedence than packets forwarded with the default PHB. Ideally, LE packets should

be forwarded only if no best-effort packet is waiting for its transmission. A straightforward implementation could be a simple priority scheduler serving the default PHB queue with higher priority than the lower-effort PHB queue. Alternative implementations may use scheduling algorithms that assign a very small weight to the LE class. This, however, may sometimes cause better service for LE packets compared to BE packets in cases when the BE share is fully utilized and the LE share not.

3. Traffic Conditioning Actions

As for most other PHBs an initial classification and marking would usually be performed at the first DS boundary node. In many cases, packets may also be pre-marked in DS aware end systems by applications due to their specific knowledge about the particular precedence of packets. There is no incentive for DS domains to distrust this initial marking, because letting LE traffic enter a DS domain causes no harm. In the worst case it evokes the same effect as it would have been marked with the default PHB, i.e., as best-effort traffic. Thus, any policing such as limiting the traffic rate is not necessary at the DS boundary.

Usually, the amount of LE traffic is implicitly limited by queueing mechanisms and related discard actions of the PHB. Therefore, there is normally no need to meter and police LE traffic explicitly.

4. Recommended DS Codepoint

The recommended codepoint for the LE PHB is 000010.

RFC 4594 [RFC4594] recommended to use CS1 as codepoint (as mentioned in [RFC3662]. This is problematic since it may cause a priority inversion resulting in treating LE packets with higher precedence than BE packets. Existing implementations SHOULD therefore use the unambiguous LE codepoint 000010 whenever possible.

5. Remarking to other DSCPs/PHBs

"DSCP bleaching", i.e., setting the DSCP to 000000 (default PHB) is not recommended for this PHB. This may cause effects that are in contrast to the original intent in protecting BE traffic from LE traffic. In case DS domains do not support the LE PHB, they may treat LE marked packets with the default PHB instead, but they should do so without remarking to the DSCP 000000. The reason for this is that later traversed DS domains may then have still the possibility to treat such packets according the LE PHB.

6. IANA Considerations

This memo includes a request to assign a Differentiated Services Field Codepoint (DSCP) 000010 from the Differentiated Services Field Codepoints (DSCP) registry https://www.iana.org/assignments/dscp-registry/dscp-registry.xml

7. Security Considerations

There are no specific security exposures for this PHB. Since it defines a new class of low forwarding priority, other traffic may be downgraded to this LE PHB in case it is remarked as LE traffic. See the general security considerations in RFC 2474 [RFC2474] and RFC 2475 [RFC2475].

8. References

8.1. 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>.

[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>.

[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>.

8.2. Informative References

[draft-bless-diffserv-lbe-phb-00] Bless, R. and K. Wehrle, "A Lower Than Best-Effort Per-Hop Behavior", draft-bless-diffserv-lbe-phb-00 (work in progress), September 1999, <https://tools.ietf.org/html/draft-bless-diffserv-lbe-phb-00>.

[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>.

[RFC3754] Bless, R. and K. Wehrle, "IP Multicast in Differentiated Services (DS) Networks", RFC 3754, DOI 10.17487/RFC3754, April 2004, <http://www.rfc-editor.org/info/rfc3754>.

[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>.

Appendix A. History of the LE PHB

A first version of this PHB was suggested by Roland Bless and Klaus Wehrle in 1999 [draft-bless-diffserv-lbe-phb-00]. After some discussion in the DiffServ Working Group Brian Carpenter and Kathie Nichols proposed a bulk handling per-domain behavior and believed a PHB was not necessary. Eventually, Lower Effort was specified as per-domain behavior and finally became [RFC3662]. More detailed information about its history can be found in Section 10 of [RFC3662].

Appendix B. Acknowledgments

Since text is borrowed from earlier Internet-Drafts and RFCs the co-authors of previous specifications are acknowledged here: Kathie Nichols and Klaus Wehrle.

Author’s Address

Roland Bless
Karlsruhe Institute of Technology (KIT)
Kaiserstr. 12
Karlsruhe
Germany

Phone: +49 721 608 46413
Email: roland.bless@kit.edu

Transport Services (tsv)
Internet-Draft
Intended status: Experimental
Expires: May 4, 2017

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

October 31, 2016

Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay
draft-briscoe-tsvwg-ecn-l4s-id-02

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, for packets carrying the L4S identifier, the network applies marking 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 throughput 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.

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 L4S marking 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.ietf-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. Note that DCTCP will still not be safe to deploy on the Internet until it satisfies the 'Safety Additions' listed in Appendix A of [I-D.briscoe-tsvwg-aqm-tcpm-rmcat-l4s-problem].

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, finance apps, remote desktop and cloud-based applications. 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], are easier to configure, because they define the queuing threshold in time not bytes, so it is invariant for different link rates. However, no matter how good the AQM, the sawtoothing rate of TCP will either cause queuing delay to vary or cause the link to be under-utilized. Even with a perfectly tuned AQM, the additional queuing delay will be of the same order as the underlying speed-of-light delay across the network. Flow-queuing can isolate one flow from another, but it cannot isolate a TCP flow from the delay variations it inflicts on itself, and it has other problems - it overrides the flow rate decisions of variable rate video

applications, it does not recognise the flows within IPSec VPN tunnels and it is relatively expensive to implement.

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.ietf-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.ietf-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", "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 (e.g. TCP Cubic, Compound, SCTP, etc).

Low-Latency, Low-Loss and Scalable (L4S) service: 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. Therefore it depends on [I-D.black-tsvwg-ecn-experimentation], which proposes to:

- update the ECN proposed standard [RFC3168] (in certain specified cases including the present document) to relax the requirement that an ECN mark must be equivalent to a drop, both when applied by the network, and when responded to by the sender;

- obsolete the experimental ECN nonce [RFC3540] (see Appendix B.1 for rationale);

- make consequent updates to the following proposed standard RFCs to reflect the above two bullets:

  * ECN for RTP [RFC6679];

  * the congestion control specifications of various DCCP CCIDs [RFC4341], [RFC4342], [RFC5622].

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:

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

- it SHOULD be common to IPv4 and IPv6 and transport agnostic;

- it SHOULD be incrementally deployable;

- it SHOULD enable an AQM to classify packets encapsulated by outer IP or lower-layer headers;
- it SHOULD consume minimal extra codepoints;
- it SHOULD not lead to some packets of a transport-layer flow being served by a different queue from others.

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.

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 against the above requirements. On the basis of this analysis, the "ECT(1) and CE codepoints" is the best compromise. Therefore this scheme is defined in detail in the following section (Section 2.2), while Appendix A has been left to document the rationale for this decision.

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 both network and host behaviour: it identifies the marking treatment that network nodes are expected to apply to L4S packets, and it identifies packets that have been sent from hosts that are expected to comply with a broad type of behaviour.

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 2.4 describes a possible 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 mark the ECN field as CE for an increasing proportion of packets, otherwise

forwarding packets unchanged as ECT(1). The L4S marking treatment is defined in Section 2.5. 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 in uncontrolled environments, 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. Pre-Requisite Transport Layer Behaviour

For a host to send packets with the L4S identifier (ECT(1)), it SHOULD implement a congestion control behaviour that ensures the flow rate is inversely proportional to the proportion of bytes in packets marked with the CE codepoint. This is termed a scalable congestion control, because the number of control signals (ECN marks) per round trip remains roughly constant for any flow rate. As with all transport behaviours, a detailed specification will need to be defined for each type of transport or application, including the timescale over which the proportionality is averaged, and control of burstiness. The inverse proportionality requirement above is worded as a 'SHOULD' rather than a 'MUST' to allow reasonable flexibility when defining these specifications.

Data Center TCP (DCTCP [I-D.ietf-tcpm-dctcp]) is an example of a scalable congestion control.

Each sender in a session can use a scalable congestion control independently of the congestion control used by the receiver(s) when they send data. Therefore theoretically there might be ECT(1) packets in one direction and ECT(0) in the other.

In general, a scalable congestion control needs feedback of the extent of CE marking on the forward path. Due to the history of TCP development, when ECN was added it reported no more than one CE mark per round trip. Some transport protocols derived from TCP mimick this behaviour while others report the extent of TCP marking. This means that some transport protocols will need to be updated as a pre-requisite for scalable congestion control. The position for a few well-known transport protocols is given below.

TCP: Support for accurate ECN feedback (AccECN [I-D.ietf-tcpm-accurate-ecn]) by both ends is a pre-requisite for scalable congestion control. However, the reverse does not apply. So even if both ends support AccECN, either of the two ends can choose not to use a scalable congestion control, whatever the other end's choice. Nonetheless, the presence of ECT(1) in the IP headers even in one direction of a TCP connection will imply that both ends support AccECN.

SCTP: An ECN feedback protocol such as that specified in [I-D.stewart-tsvwg-sctpecn] would be a pre-requisite for scalable congestion control. That draft would update the ECN feedback protocol sketched out in Appendix A of the standards track specification of SCTP [RFC4960] by adding a field to report the number of CE marks.

RTP over UDP: A pre-requisite for scalable congestion control is for both (all) ends of one media-level hop to signal ECN support using the ecn-capable-rtp attribute [RFC6679]. However, the reverse does not apply, so each end of a media-level hop can independently choose not to use a scalable congestion control, even if both ends support ECN. Nonetheless, the presence of ECT(1) implies that both (all) ends of that hop support ECN.

DCCP: The ACK vector in DCCP [RFC4340] is already sufficient to report the extent of CE marking as needed by a scalable congestion control.

2.4. L4S Packet Identification by Network Nodes with Transport-Layer Awareness

To implement the L4S treatment, a network node does not need to identify transport-layer flows. Nonetheless, if an implementer is willing to identify transport-layer flows at a network node, and if the most recent ECT packet in the same flow was ECT(0), the node MAY classify CE packets for classic ECN [RFC3168] treatment. In all other cases, a network node MUST classify CE packets for L4S treatment. Examples of such other cases are: i) if no ECT packets have yet been identified in a flow; ii) if it is not desirable for a network node to identify transport-layer flows; or iii) if the most recent ECT packet in a flow was ECT(1).

If an implementer uses flow-awareness to classify CE packets, to determine whether the flow is using ECT(0) or ECT(1) it only uses the most recent ECT packet of a flow {ToDo: this advice will need to be verified experimentally}. This is 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. A host ought to be able to detect such a change from a change in RTT variation.

2.5. The Meaning of CE Relative to Drop

The likelihood that an AQM drops a Not-ECT Classic packet (p_C) MUST be roughly proportional to the square of the likelihood that it would have marked it if it had been an L4S packet (p_L). That is

$$p\_C \tilde{=} (p\_L / k)^2$$

The constant of proportionality (k) does not have to be standardised for interoperability, but a value of 2 is RECOMMENDED.

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

The term 'likelihood' is used above to allow for marking and dropping to be either probabilistic or deterministic. The example AQMs in [I-D.briscoe-aqm-dualq-coupled] drop and mark 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 an 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.

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 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 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.bagnulo-tswg-generalized-ecn] Bagnulo, M. and B. Briscoe, "Adding Explicit Congestion Notification (ECN) to TCP control packets", draft-bagnulo-tswg-generalized-ecn-00 (work in progress), July 2016.

[I-D.black-tsvwg-ecn-experimentation]
Black, D., "Explicit Congestion Notification (ECN) Experimentation", draft-black-tsvwg-ecn-experimentation-02 (work in progress), October 2016.

[I-D.briscoe-aqm-dualq-coupled]
Schepper, K., Briscoe, B., Bondarenko, O., and I. Tsang, "DualQ Coupled AQM for Low Latency, Low Loss and Scalable Throughput", draft-briscoe-aqm-dualq-coupled-01 (work in progress), March 2016.

[I-D.briscoe-tsvwg-aqm-tcpm-rmcat-l4s-problem]
Briscoe, B., Schepper, K., and M. Bagnulo, "Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Problem Statement", draft-briscoe-tsvwg-aqm-tcpm-rmcat-l4s-problem-02 (work in progress), July 2016.

[I-D.ietf-aqm-fq-codel]
Hoeiland-Joergensen, T., McKenney, P., dave.taht@gmail.com, d., Gettys, J., and E. Dumazet, "The FlowQueue-CoDel Packet Scheduler and Active Queue Management Algorithm", draft-ietf-aqm-fq-codel-06 (work in progress), March 2016.

[I-D.ietf-aqm-pie]
Pan, R., Natarajan, P., Baker, F., and G. White, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem", draft-ietf-aqm-pie-10 (work in progress), September 2016.

[I-D.ietf-tcpm-accurate-ecn]
Briscoe, B., Kuehlewind, M., and R. Scheffenegger, "More Accurate ECN Feedback in TCP", draft-ietf-tcpm-accurate-ecn-02 (work in progress), October 2016.

[I-D.ietf-tcpm-cubic]
Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", draft-ietf-tcpm-cubic-02 (work in progress), August 2016.

[I-D.ietf-tcpm-dctcp]
Bensley, S., Eggert, L., Thaler, D., Balasubramanian, P., and G. Judd, "Datacenter TCP (DCTCP): TCP Congestion Control for Datacenters", draft-ietf-tcpm-dctcp-02 (work in progress), July 2016.

[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-07 (work in progress), July 2016.

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

[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 D2.3; Appendix C.2, August 2015, <https://riteproject.files.wordpress.com/2015/12/rite-deliverable-2-3.pdf>.

[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>.

[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>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

[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>.

[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>.

[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>.

[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>.

[RFC7713] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements", RFC 7713, DOI 10.17487/RFC7713, December 2015, <http://www.rfc-editor.org/info/rfc7713>.

[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;

* 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 ought not to 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, 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. To address this problem, [I-D.bagnulo-tswg-generalized-ecn] proposes an experiment in which all TCP control packets and retransmissions are ECN-capable.

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;

* 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 above, 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 above), 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;

* ECN codepoints would not be used for classic [RFC3168] 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 [RFC3168] ECN deployment in any network nodes;

B. it was generally agreed that there was little chance of any client devices being deployed with classic [RFC3168] TCP-ECN on by default (note that classic TCP-ECN is already on-by-default on many servers);

C. for TCP connections, developers of client OSs would all have to agree not to encourage further deployment of classic ECN. Specifically, at the start of a TCP connection classic ECN could be disabled during negotation of the ECN capability:

   + an L4S-capable host would have to disable ECN if the corresponding host did not support accurate ECN feedback [RFC7560], which is a prerequisite for the L4S service;

   + developers of operating systems for user devices would only enable ECN by default for TCP once the stack implemented L4S and accurate ECN feedback [RFC7560] including requesting accurate ECN feedback by default.

Cons:

Near-infeasible deployment constraints: The constraints for deployment above represent a highly unlikely, but not completely impossible, set of circumstances. 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:

- A new protocol ID would need to be paired with the old one for each transport (TCP, SCTP, UDP, etc.);
- 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;
- 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;
- 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(1) + 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 summarises the text and is not meant to be understandable without having read 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 as the ECN nonce [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 of losses if retransmissions are not necessary or available otherwise). [RFC3540] proposes that a TCP sender could set either of ECT(0) or ECT(1) in each packet of a flow and remember the sequence it had set, 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 in the base IP header. For instance:

- 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 [RFC7560] intended for L4S;

- A network can enforce a congestion response to its ECN markings (or packet losses) by auditing congestion exposure (ConEx) [RFC7713]. 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. 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: consider Jonathan Morton's Explicit Load Regulation (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
Nokia Bell Labs
Antwerp
Belgium

Email: koen.de_schepper@nokia.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
Nokia Bell Labs
Antwerp
Belgium

Email: ing-jyh.tsang@nokia.com

Transport Area Working Group
Internet-Draft
Updates: 6040, 2661, 1701, 2784, 2637, 3931 (if approved)
Intended status: Standards Track
Expires: January 9, 2017

B. Briscoe
Simula Research Laboratory

July 8, 2016

Propagating Explicit Congestion Notification Across IP Tunnel Headers Separated by a Shim
draft-briscoe-tsvwg-rfc6040bis-01

Abstract

RFC 6040 on "Tunnelling of Explicit Congestion Notification" made the rules for propagation of ECN consistent for all forms of IP in IP tunnel. This specification extends the scope of RFC 6040 to include tunnels where two IP headers are separated by a shim header that cannot stand alone.

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.

Copyright Notice

Table of Contents

1. Scope of RFC 6040

RFC 6040 on "Tunnelling of Explicit Congestion Notification" [RFC6040] made the rules for propagation of Explicit Congestion Notification (ECN [RFC3168]) consistent for all forms of IP in IP tunnel. The scope of RFC 6040 was stated as

> "...ECN field processing at encapsulation and decapsulation for any IP-in-IP tunnelling, whether IPsec or non-IPsec tunnels. It applies irrespective of whether IPv4 or IPv6 is used for either the inner or outer headers. ..."

A common pattern for many tunnelling protocols is to encapsulate an inner IP header with shim header(s) then an outer IP header. To clear up confusion, this specification clarifies that the scope of RFC 6040 includes any IP-in-IP tunnel, including those with shim header(s) between the IP headers.

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 RFC 2119 [RFC2119].

3. IP-in-IP Tunnels with Tightly Coupled Shim Headers

In many cases the shim header(s) and the outer IP header are always added (or removed) as part of the same process. We call this a tightly coupled shim header. Processing the shim and outer together is often necessary because the shim(s) are not sufficient for packet forwarding in their own right; not unless complemented by an outer header.

For all such tightly coupled shim headers, the rules in [RFC6040] for propagating the ECN field SHOULD be applied directly between the inner and outer IP headers. This specification therefore updates the following specifications of tightly coupled shim headers by adding that RFC 6040 SHOULD apply when the shim header is used between IP headers:

- L2TPv2 [RFC2661], L2TPv3 [RFC3931]
- GRE [RFC1701], [RFC2784]
- PPTP [RFC2637]
- GTP [GTPv1], [GTPv1-U], [GTPv2-C]
- VXLAN [RFC7348].

Geneve [I-D.ietf-nvo3-geneve] and Generic UDP Encapsulation (GUE) [I-D.ietf-nvo3-gue] are also tightly coupled shim headers, but their specifications already refer to RFC 6040 for ECN encapsulation.

The above is written as a 'SHOULD' not a 'MUST' to allow for the possibility that the structure of some pre-existing tunnel implementations might make it hard to predict what other headers will be added or removed subsequently.

Although the definition of the various GTP shim headers is under the control of the 3GPP, it is hard to determine whether the 3GPP or the IETF controls standardization of the _process_ of adding both a GTP and an IP header to an inner IP header. Nonetheless, the present specification is provided so that the 3GPP can refer to it from any of its own specifications of GTP and IP header processing.

Similarly, VXLAN is not under the control of the IETF, but the present specification is provided so that the authors of any future update to the VXLAN specification can refer to it.

More generally, whatever form IP-in-IP tunnelling takes, the ECN field SHOULD be propagated according to the rules in RFC 6040 wherever possible. Otherwise [I-D.ietf-tsvwg-ecn-encap-guidelines] gives more general guidance on how to propagate ECN to and from protocols that encapsulate IP.

4. IANA Considerations (to be removed by RFC Editor)

This memo includes no request to IANA.

5. Security Considerations

The Security Considerations in RFC 6040 apply equally to the wider scope defined by the present specification.

6. Comments Solicited

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.

7. Normative References

[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-nvo3-geneve] Gross, J. and I. Ganga, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-01 (work in progress), January 2016.

[I-D.ietf-nvo3-gue] Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-nvo3-gue-04 (work in progress), July 2016.

[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-06 (work in progress), July 2016.

[RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, DOI 10.17487/RFC1701, October 1994, <http://www.rfc-editor.org/info/rfc1701>.

[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>.

[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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc2784>.

[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>.

[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>.

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

[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>.

Author’s Address

Bob Briscoe
Simula Research Laboratory
UK

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

AVTCORE Working Group
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: February 19, 2017

C. Perkins
University of Glasgow
V. Singh
callstats.io

August 18, 2016

Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions
draft-ietf-avtcore-rtp-circuit-breakers-18

Abstract

The Real-time Transport Protocol (RTP) is widely used in telephony, video conferencing, and telepresence applications. Such applications are often run on best-effort UDP/IP networks. If congestion control is not implemented in these applications, then network congestion can lead to uncontrolled packet loss, and a resulting deterioration of the user's multimedia experience. The congestion control algorithm acts as a safety measure, stopping RTP flows from using excessive resources, and protecting the network from overload. At the time of this writing, however, while there are several proprietary solutions, there is no standard algorithm for congestion control of interactive RTP flows.

This document does not propose a congestion control algorithm. It instead defines a minimal set of RTP circuit breakers: conditions under which an RTP sender needs to stop transmitting media data, to protect the network from excessive congestion. It is expected that, in the absence of long-lived excessive congestion, RTP applications running on best-effort IP networks will be able to operate without triggering these circuit breakers. To avoid triggering the RTP circuit breaker, any standards-track congestion control algorithms defined for RTP will need to operate within the envelope set by these RTP circuit breaker algorithms.

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 February 19, 2017.

Table of Contents

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is widely used in voice-over-IP, video teleconferencing, and telepresence systems. Many of these systems run over best-effort UDP/IP networks, and can

suffer from packet loss and increased latency if network congestion occurs. Designing effective RTP congestion control algorithms, to adapt the transmission of RTP-based media to match the available network capacity, while also maintaining the user experience, is a difficult but important problem. Many such congestion control and media adaptation algorithms have been proposed, but to date there is no consensus on the correct approach, or even that a single standard algorithm is desirable.

This memo does not attempt to propose a new RTP congestion control algorithm. Instead, we propose a small set of RTP circuit breakers: mechanisms that terminate RTP flows in conditions under which there is general agreement that serious network congestion is occurring. The RTP circuit breakers proposed in this memo are a specific instance of the general class of network transport circuit breakers [I-D.ietf-tsvwg-circuit-breaker], designed to act as a protection mechanism of last resort to avoid persistent excessive congestion. To avoid triggering the RTP circuit breaker, any standards-track congestion control algorithms defined for RTP will need to operate within the envelope set by the RTP circuit breaker algorithms defined by this memo.

2. Background

We consider congestion control for unicast RTP traffic flows. This is the problem of adapting the transmission of an audio/visual data flow, encapsulated within an RTP transport session, from one sender to one receiver, so that it does not use more capacity than is available along the network path. Such adaptation needs to be done in a way that limits the disruption to the user experience caused by both packet loss and excessive rate changes. Congestion control for multicast flows is outside the scope of this memo. Multicast traffic needs different solutions, since the available capacity estimator for a group of receivers will differ from that for a single receiver, and because multicast congestion control has to consider issues of fairness across groups of receivers that do not apply to unicast flows.

Congestion control for unicast RTP traffic can be implemented in one of two places in the protocol stack. One approach is to run the RTP traffic over a congestion controlled transport protocol, for example over TCP, and to adapt the media encoding to match the dictates of the transport-layer congestion control algorithm. This is safe for the network, but can be suboptimal for the media quality unless the transport protocol is designed to support real-time media flows. We do not consider this class of applications further in this memo, as their network safety is guaranteed by the underlying transport.

Alternatively, RTP flows can be run over a non-congestion controlled transport protocol, for example UDP, performing rate adaptation at the application layer based on RTP Control Protocol (RTCP) feedback. With a well-designed, network-aware, application, this allows highly effective media quality adaptation, but there is potential to cause persistent congestion in the network if the application does not adapt its sending rate in a timely and effective manner. We consider this class of applications in this memo.

Congestion control relies on monitoring the delivery of a media flow, and responding to adapt the transmission of that flow when there are signs that the network path is congested. Network congestion can be detected in one of three ways: 1) a receiver can infer the onset of congestion by observing an increase in one-way delay caused by queue build-up within the network; 2) if Explicit Congestion Notification (ECN) [RFC3168] is supported, the network can signal the presence of congestion by marking packets using ECN Congestion Experienced (CE) marks (this could potentially be augmented by mechanisms such as ConEX [RFC7713], or other future protocol extensions for network signalling of congestion); or 3) in the extreme case, congestion will cause packet loss that can be detected by observing a gap in the received RTP sequence numbers.

Once the onset of congestion is observed, the receiver has to send feedback to the sender to indicate that the transmission rate needs to be reduced. How the sender reduces the transmission rate is highly dependent on the media codec being used, and is outside the scope of this memo.

There are several ways in which a receiver can send feedback to a media sender within the RTP framework:

- The base RTP specification [RFC3550] defines RTCP Reception Report (RR) packets to convey reception quality feedback information, and Sender Report (SR) packets to convey information about the media transmission. RTCP SR packets contain data that can be used to reconstruct media timing at a receiver, along with a count of the total number of octets and packets sent. RTCP RR packets report on the fraction of packets lost in the last reporting interval, the cumulative number of packets lost, the highest sequence number received, and the inter-arrival jitter. The RTCP RR packets also contain timing information that allows the sender to estimate the network round trip time (RTT) to the receivers. RTCP reports are sent periodically, with the reporting interval being determined by the number of SSRCs used in the session and a configured session bandwidth estimate (the number of synchronisation sources (SSRCs) used is usually two in a unicast session, one for each participant, but can be greater if the participants send multiple

media streams). The interval between reports sent from each receiver tends to be on the order of a few seconds on average, although it varies with the session bandwidth, and sub-second reporting intervals are possible in high bandwidth sessions, and it is randomised to avoid synchronisation of reports from multiple receivers. RTCP RR packets allow a receiver to report ongoing network congestion to the sender. However, if a receiver detects the onset of congestion part way through a reporting interval, the base RTP specification contains no provision for sending the RTCP RR packet early, and the receiver has to wait until the next scheduled reporting interval.

- The RTCP Extended Reports (XR) [RFC3611] allow reporting of more complex and sophisticated reception quality metrics, but do not change the RTCP timing rules. RTCP extended reports of potential interest for congestion control purposes are the extended packet loss, discard, and burst metrics [RFC3611], [RFC7002], [RFC7097], [RFC7003], [RFC6958]; and the extended delay metrics [RFC6843], [RFC6798]. Other RTCP Extended Reports that could be helpful for congestion control purposes might be developed in future.

- Rapid feedback about the occurrence of congestion events can be achieved using the Extended RTP Profile for RTCP-Based Feedback (RTP/AVPF) [RFC4585] (or its secure variant, RTP/SAVPF [RFC5124]) in place of the RTP/AVP profile [RFC3551]. This modifies the RTCP timing rules to allow RTCP reports to be sent early, in some cases immediately, provided the RTCP transmission rate keeps within its bandwidth allocation. It also defines transport-layer feedback messages, including negative acknowledgements (NACKs), that can be used to report on specific congestion events. RTP Codec Control Messages [RFC5104] extend the RTP/AVPF profile with additional feedback messages that can be used to influence that way in which rate adaptation occurs, but do not further change the dynamics of how rapidly feedback can be sent. Use of the RTP/AVPF profile is dependent on signalling.

- Finally, Explicit Congestion Notification (ECN) for RTP over UDP [RFC6679] can be used to provide feedback on the number of packets that received an ECN Congestion Experienced (CE) mark. This RTCP extension builds on the RTP/AVPF profile to allow rapid congestion feedback when ECN is supported.

In addition to these mechanisms for providing feedback, the sender can include an RTP header extension in each packet to record packet transmission times [RFC5450]. Accurate transmission timestamps can be helpful for estimating queuing delays, to get an early indication of the onset of congestion.

Taken together, these various mechanisms allow receivers to provide feedback on the senders when congestion events occur, with varying degrees of timeliness and accuracy. The key distinction is between systems that use only the basic RTCP mechanisms, without RTP/AVPF rapid feedback, and those that use the RTP/AVPF extensions to respond to congestion more rapidly.

3. 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]. This interpretation of these key words applies only when written in ALL CAPS. Mixed- or lower-case uses of these key words are not to be interpreted as carrying special significance in this memo.

The definition of the RTP circuit breaker is specified in terms of the following variables:

- Td is the deterministic RTCP reporting interval, as defined in Section 6.3.1 of [RFC3550].

- Tdr is the sender’s estimate of the deterministic RTCP reporting interval, Td, calculated by a receiver of the data it is sending. Tdr is not known at the sender, but can be estimated by executing the algorithm in Section 6.2 of [RFC3550] using the average RTCP packet size seen at the sender, the number of members reported in the receiver’s SR/RR report blocks, and whether the receiver is sending SR or RR packets. Tdr is recalculated when each new RTCP SR/RR report is received, but the media timeout circuit breaker (see Section 4.2) is only reconsidered when Tdr increases.

- Tr is the network round-trip time, calculated by the sender using the algorithm in Section 6.4.1 of [RFC3550] and smoothed using an exponentially weighted moving average as Tr = (0.8 * Tr) + (0.2 * Tr_new) where Tr_new is the latest RTT estimate obtained from an RTCP report. The weight is chosen so old estimates decay over k intervals.

- k is the non-reporting threshold (see Section 4.2).

- Tf is the media framing interval at the sender. For applications sending at a constant frame rate, Tf is the inter-frame interval. For applications that switch between a small set of possible frame rates, for example when sending speech with comfort noise, where comfort noise frames are sent less often than speech frames, Tf is set to the longest of the inter-frame intervals of the different frame rates. For applications that send periodic frames but

dynamically vary their frame rate, Tf is set to the largest inter-frame interval used in the last 10 seconds. For applications that send less than one frame every 10 seconds, or that have no concept of periodic frames (e.g., text conversation [RFC4103], or pointer events [RFC2862]), Tf is set to the time interval since the previous frame when each frame is sent.

- G is the frame group size. That is, the number of frames that are coded together based on a particular sending rate setting. If the codec used by the sender can change its rate on each frame, G = 1; otherwise G is set to the number of frames before the codec can adjust to the new rate. For codecs that have the concept of a group-of-pictures (GoP), G is likely the GoP length.

- T_rr_interval is the minimal interval between RTCP reports, as defined in Section 3.4 of [RFC4585]; it is only meaningful for implementations of RTP/AVPF profile [RFC4585] or the RTP/SAVPF profile [RFC5124].

- X is the estimated throughput a TCP connection would achieve over a path, in bytes per second.

- s is the size of RTP packets being sent, in bytes. If the RTP packets being sent vary in size, then the average size over the packet comprising the last 4 * G frames MUST be used (this is intended to be comparable to the four loss intervals used in [RFC5348]).

- p is the loss event rate, between 0.0 and 1.0, that would be seen by a TCP connection over a particular path. When used in the RTP congestion circuit breaker, this is approximated as described in Section 4.3.

- t_RTO is the retransmission timeout value that would be used by a TCP connection over a particular path, in seconds. This MUST be approximated using t_RTO = 4 * Tr when used as part of the RTP congestion circuit breaker.

- b is the number of packets that are acknowledged by a single TCP acknowledgement. Following [RFC5348], it is RECOMMENDED that the value b = 1 is used as part of the RTP congestion circuit breaker.

4. RTP Circuit Breakers for Systems Using the RTP/AVP Profile

The feedback mechanisms defined in [RFC3550] and available under the RTP/AVP profile [RFC3551] are the minimum that can be assumed for a baseline circuit breaker mechanism that is suitable for all unicast applications of RTP. Accordingly, for an RTP circuit breaker to be

useful, it needs to be able to detect that an RTP flow is causing excessive congestion using only basic RTCP features, without needing RTCP XR feedback or the RTP/AVPF profile for rapid RTCP reports.

RTCP is a fundamental part of the RTP protocol, and the mechanisms described here rely on the implementation of RTCP. Implementations that claim to support RTP, but that do not implement RTCP, will be unable to use the circuit breaker mechanisms described in this memo. Such implementations SHOULD NOT be used on networks that might be subject to congestion unless equivalent mechanisms are defined using some non-RTCP feedback channel to report congestion and signal circuit breaker conditions.

The RTCP timeout circuit breaker (Section 4.1) will trigger if an implementation of this memo attempts to interwork with an endpoint that does not support RTCP. Implementations that sometimes need to interwork with endpoints that do not support RTCP need to disable the RTP circuit breakers if they don’t receive some confirmation via signalling that the remote endpoint implements RTCP (the presence of an SDP "a=rtcp:" attribute in an answer might be such an indication). The RTP Circuit Breaker SHOULD NOT be disabled on networks that might be subject to congestion, unless equivalent mechanisms are defined using some non-RTCP feedback channel to report congestion and signal circuit breaker conditions [I-D.ietf-tsvwg-circuit-breaker].

Three potential congestion signals are available from the basic RTCP SR/RR packets and are reported for each SSRC in the RTP session:

1. The sender can estimate the network round-trip time once per RTCP reporting interval, based on the contents and timing of RTCP SR and RR packets.

2. Receivers report a jitter estimate (the statistical variance of the RTP data packet inter-arrival time) calculated over the RTCP reporting interval. Due to the nature of the jitter calculation ([RFC3550], section 6.4.4), the jitter is only meaningful for RTP flows that send a single data packet for each RTP timestamp value (i.e., audio flows, or video flows where each packet comprises one video frame).

3. Receivers report the fraction of RTP data packets lost during the RTCP reporting interval, and the cumulative number of RTP packets lost over the entire RTP session.

These congestion signals limit the possible circuit breakers, since they give only limited visibility into the behaviour of the network.

RTT estimates are widely used in congestion control algorithms, as a proxy for queuing delay measures in delay-based congestion control or to determine connection timeouts. RTT estimates derived from RTCP SR and RR packets sent according to the RTP/AVP timing rules are too infrequent to be useful for congestion control, and don't give enough information to distinguish a delay change due to routing updates from queuing delay caused by congestion. Accordingly, we cannot use the RTT estimate alone as an RTP circuit breaker.

Increased jitter can be a signal of transient network congestion, but in the highly aggregated form reported in RTCP RR packets, it offers insufficient information to estimate the extent or persistence of congestion. Jitter reports are a useful early warning of potential network congestion, but provide an insufficiently strong signal to be used as a circuit breaker.

The remaining congestion signals are the packet loss fraction and the cumulative number of packets lost. If considered carefully, and over an appropriate time frame to distinguish transient problems from long term issues [I-D.ietf-tsvwg-circuit-breaker], these can be effective indicators that persistent excessive congestion is occurring in networks where packet loss is primarily due to queue overflows, although loss caused by non-congestive packet corruption can distort the result in some networks. TCP congestion control [RFC5681] intentionally tries to fill the router queues, and uses the resulting packet loss as congestion feedback. An RTP flow competing with TCP traffic will therefore expect to see a non-zero packet loss fraction, and some variation in queuing latency, in normal operation when sharing a path with other flows, that needs to be accounted for when determining the circuit breaker threshold [I-D.ietf-tsvwg-circuit-breaker]. This behaviour of TCP is reflected in the congestion circuit breaker below, and will affect the design of any RTP congestion control protocol.

Two packet loss regimes can be observed: 1) RTCP RR packets show a non-zero packet loss fraction, while the extended highest sequence number received continues to increment; and 2) RR packets show a loss fraction of zero, but the extended highest sequence number received does not increment even though the sender has been transmitting RTP data packets. The former corresponds to the TCP congestion avoidance state, and indicates a congested path that is still delivering data; the latter corresponds to a TCP timeout, and is most likely due to a path failure. A third condition is that data is being sent but no RTCP feedback is received at all, corresponding to a failure of the reverse path. We derive circuit breaker conditions for these loss regimes in the following.

4.1. RTP/AVP Circuit Breaker #1: RTCP Timeout

An RTCP timeout can occur when RTP data packets are being sent, but there are no RTCP reports returned from the receiver. This is either due to a failure of the receiver to send RTCP reports, or a failure of the return path that is preventing those RTCP reporting from being delivered. In either case, it is not safe to continue transmission, since the sender has no way of knowing if it is causing congestion.

An RTP sender that has not received any RTCP SR or RTCP RR packets reporting on the SSRC it is using, for a time period of at least three times its deterministic RTCP reporting interval, Td, without the randomization factor, and using the fixed minimum interval of Tmin=5 seconds, SHOULD cease transmission (see Section 4.5). The rationale for this choice of timeout is as described in Section 6.2 of [RFC3550] ("so that implementations which do not use the reduced value for transmitting RTCP packets are not timed out by other participants prematurely"), as updated by Section 6.1.4 of [I-D.ietf-avtcore-rtp-multi-stream] to account for the use of the RTP/AVPF profile [RFC4585] or the RTP/SAVPF profile [RFC5124].

To reduce the risk of premature timeout, implementations SHOULD NOT configure the RTCP bandwidth such that Td is larger than 5 seconds. Similarly, implementations that use the RTP/AVPF profile [RFC4585] or the RTP/SAVPF profile [RFC5124] SHOULD NOT configure T_rr_interval to values larger than 4 seconds (the reduced limit for T_rr_interval follows Section 6.1.3 of [I-D.ietf-avtcore-rtp-multi-stream]).

The choice of three RTCP reporting intervals as the timeout is made following Section 6.3.5 of RFC 3550 [RFC3550]. This specifies that participants in an RTP session will timeout and remove an RTP sender from the list of active RTP senders if no RTP data packets have been received from that RTP sender within the last two RTCP reporting intervals. Using a timeout of three RTCP reporting intervals is therefore large enough that the other participants will have timed out the sender if a network problem stops the data packets it is sending from reaching the receivers, even allowing for loss of some RTCP packets.

If a sender is transmitting a large number of RTP media streams, such that the corresponding RTCP SR or RR packets are too large to fit into the network MTU, the receiver will generate RTCP SR or RR packets in a round-robin manner. In this case, the sender SHOULD treat receipt of an RTCP SR or RR packet corresponding to any SSRC it sent on the same 5-tuple of source and destination IP address, port, and protocol, as an indication that the receiver and return path are working, preventing the RTCP timeout circuit breaker from triggering.

4.2. RTP/AVP Circuit Breaker #2: Media Timeout

If RTP data packets are being sent, but the RTCP SR or RR packets reporting on that SSRC indicate a non-increasing extended highest sequence number received, this is an indication that those RTP data packets are not reaching the receiver. This could be a short-term issue affecting only a few RTP packets, perhaps caused by a slow to open firewall or a transient connectivity problem, but if the issue persists, it is a sign of a more ongoing and significant problem (a "media timeout").

The time needed to declare a media timeout depends on the parameters Tdr, Tr, Tf, and on the non-reporting threshold k. The value of k is chosen so that when Tdr is large compared to Tr and Tf, receipt of at least k RTCP reports with non-increasing extended highest sequence number received gives reasonable assurance that the forward path has failed, and that the RTP data packets have not been lost by chance. The RECOMMENDED value for k is 5 reports.

When Tdr < Tf, then RTP data packets are being sent at a rate less than one per RTCP reporting interval of the receiver, so the extended highest sequence number received can be expected to be non-increasing for some receiver RTCP reporting intervals. Similarly, when Tdr < Tr, some receiver RTCP reporting intervals might pass before the RTP data packets arrive at the receiver, also leading to reports where the extended highest sequence number received is non-increasing. Both issues require the media timeout interval to be scaled relative to the threshold, k.

The media timeout RTP circuit breaker is therefore as follows. When starting sending, calculate MEDIA_TIMEOUT using:

```
MEDIA_TIMEOUT = ceil(k * max(Tf, Tr, Tdr) / Tdr)
```

When a sender receives an RTCP packet that indicates reception of the media it has been sending, then it cancels the media timeout circuit breaker. If it is still sending, then it MUST calculate a new value for MEDIA_TIMEOUT, and set a new media timeout circuit breaker.

If a sender receives an RTCP packet indicating that its media was not received, it MUST calculate a new value for MEDIA_TIMEOUT. If the new value is larger than the previous, it replaces MEDIA_TIMEOUT with the new value, extending the media timeout circuit breaker; otherwise it keeps the original value of MEDIA_TIMEOUT. This process is known as reconsidering the media timeout circuit breaker.

If MEDIA_TIMEOUT consecutive RTCP packets are received indicating that the media being sent was not received, and the media timeout

circuit breaker has not been cancelled, then the media timeout circuit breaker triggers. When the media timeout circuit breaker triggers, the sender SHOULD cease transmission (see Section 4.5).

When stopping sending an RTP stream, a sender MUST cancel the corresponding media timeout circuit breaker.

4.3. RTP/AVP Circuit Breaker #3: Congestion

If RTP data packets are being sent, and the corresponding RTCP SR or RR packets show non-zero packet loss fraction and increasing extended highest sequence number received, then those RTP data packets are arriving at the receiver, but some degree of congestion is occurring. The RTP/AVP profile [RFC3551] states that:

> If best-effort service is being used, RTP receivers SHOULD monitor packet loss to ensure that the packet loss rate is within acceptable parameters. Packet loss is considered acceptable if a TCP flow across the same network path and experiencing the same network conditions would achieve an average throughput, measured on a reasonable time scale, that is not less than the throughput the RTP flow is achieving. This condition can be satisfied by implementing congestion control mechanisms to adapt the transmission rate (or the number of layers subscribed for a layered multicast session), or by arranging for a receiver to leave the session if the loss rate is unacceptably high.
>
> The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in time scale and throughput. The time scale on which TCP throughput is measured is the round-trip time of the connection. In essence, this requirement states that it is not acceptable to deploy an application (using RTP or any other transport protocol) on the best-effort Internet which consumes bandwidth arbitrarily and does not compete fairly with TCP within an order of magnitude.

The phase "order of magnitude" in the above means within a factor of ten, approximately. In order to implement this, it is necessary to estimate the throughput a bulk TCP connection would achieve over the path. For a long-lived TCP Reno connection, it has been shown that the TCP throughput, X, in bytes per second, can be estimated using [Padhye]:

$$X = \frac{s}{Tr*\sqrt{2*b*p/3}+(t\_RTO * (3*\sqrt{3*b*p/8}) * p * (1+32*p*p))}$$

This is the same approach to estimated TCP throughput that is used in [RFC5348]. Under conditions of low packet loss the second term on the denominator is small, so this formula can be approximated with reasonable accuracy as follows [Mathis]:

$$X = \frac{s}{Tr*\sqrt{2*b*p/3}}$$

It is RECOMMENDED that this simplified throughput equation be used, since the reduction in accuracy is small, and it is much simpler to calculate than the full equation. Measurements have shown that the simplified TCP throughput equation is effective as an RTP circuit breaker for multimedia flows sent to hosts on residential networks using ADSL and cable modem links [Singh]. The data shows that the full TCP throughput equation tends to be more sensitive to packet loss and triggers the RTP circuit breaker earlier than the simplified equation. Implementations that desire this extra sensitivity MAY use the full TCP throughput equation in the RTP circuit breaker. Initial measurements in LTE networks have shown that the extra sensitivity is helpful in that environment, with the full TCP throughput equation giving a more balanced circuit breaker response than the simplified TCP equation [Sarker]; other networks might see similar behaviour.

No matter what TCP throughput equation is chosen, two parameters need to be estimated and reported to the sender in order to calculate the throughput: the round trip time, Tr, and the loss event rate, p (the packet size, s, is known to the sender). The round trip time can be estimated from RTCP SR and RR packets. This is done too infrequently for accurate statistics, but is the best that can be done with the standard RTCP mechanisms.

Report blocks in RTCP SR or RR packets contain the packet loss fraction, rather than the loss event rate, so p cannot be reported (TCP typically treats the loss of multiple packets within a single RTT as one loss event, but RTCP RR packets report the overall fraction of packets lost, and does not report when the packet losses occurred). Using the loss fraction in place of the loss event rate can overestimate the loss. We believe that this overestimate will not be significant, given that we are only interested in order of magnitude comparison ([Floyd] section 3.2.1 shows that the difference is small for steady-state conditions and random loss, but using the loss fraction is more conservative in the case of bursty loss).

The congestion circuit breaker is therefore: when a sender that is transmitting at least one RTP packet every max(Tdr, Tr) seconds receives an RTCP SR or RR packet that contains a report block for an SSRC it is using, the sender MUST record the value of the fraction

lost field from the report block, and the time since the last report block was received, for that SSRC. If more than CB_INTERVAL (see below) report blocks have been received for that SSRC, the sender MUST calculate the average fraction lost over the last CB_INTERVAL reporting intervals, and then estimate the TCP throughput that would be achieved over the path using the chosen TCP throughput equation and the measured values of the round-trip time, Tr, the loss event rate, p (approximated by the average fraction lost, as is described below), and the packet size, s. The estimate of the TCP throughput, X, is then compared with the actual sending rate of the RTP stream. If the actual sending rate of the RTP stream is more than 10 * X, then the congestion circuit breaker is triggered.

The average fraction lost is calculated based on the sum, over the last CB_INTERVAL reporting intervals, of the fraction lost in each reporting interval multiplied by the duration of the corresponding reporting interval, divided by the total duration of the last CB_INTERVAL reporting intervals. The CB_INTERVAL parameter is set to:

```
CB_INTERVAL =
   ceil(3*min(max(10*G*Tf, 10*Tr, 3*Tdr), max(15, 3*Td))/(3*Tdr))
```

The parameters that feed into CB_INTERVAL are chosen to give the congestion control algorithm time to react to congestion. They give at least three RTCP reports, ten round trip times, and ten groups of frames to adjust the rate to reduce the congestion to a reasonable level. It is expected that a responsive congestion control algorithm will begin to respond with the next group of frames after it receives indication of congestion, so CB_INTERVAL ought to be a much longer interval than the congestion response.

If the RTP/AVPF profile [RFC4585] or the RTP/SAVPF [RFC5124] is used, and the T_rr_interval parameter is used to reduce the frequency of regular RTCP reports, then the value Tdr in the above expression for the CB_INTERVAL parameter MUST be replaced by max(T_rr_interval, Tdr).

The CB_INTERVAL parameter is calculated on joining the session, and recalculated on receipt of each RTCP packet, after checking whether the media timeout circuit breaker or the congestion circuit breaker has been triggered.

To ensure a timely response to persistent congestion, implementations SHOULD NOT configure the RTCP bandwidth such that Tdr is larger than 5 seconds. Similarly, implementations that use the RTP/AVPF profile [RFC4585] or the RTP/SAVPF profile [RFC5124] SHOULD NOT configure T_rr_interval to values larger than 4 seconds (the reduced limit for

T_rr_interval follows Section 6.1.3 of
[I-D.ietf-avtcore-rtp-multi-stream]).

The rationale for enforcing a minimum sending rate below which the congestion circuit breaker will not trigger is to avoid spurious circuit breaker triggers when the number of packets sent per RTCP reporting interval is small, and hence the fraction lost samples are subject to measurement artefacts. The bound of at least one packet every max(Tdr, Tr) seconds is derived from the one packet per RTT minimum sending rate of TCP [RFC5405], adapted for use with RTP where the RTCP reporting interval is decoupled from the network RTT.

When the congestion circuit breaker is triggered, the sender SHOULD cease transmission (see Section 4.5). However, if the sender is able to reduce its sending rate by a factor of (approximately) ten, then it MAY first reduce its sending rate by this factor (or some larger amount) to see if that resolves the congestion. If the sending rate is reduced in this way and the congestion circuit breaker triggers again after the next CB_INTERVAL RTCP reporting intervals, the sender MUST then cease transmission. An example of such a rate reduction might be a video conferencing system that backs off to sending audio only, before completely dropping the call. If such a reduction in sending rate resolves the congestion problem, the sender MAY gradually increase the rate at which it sends data after a reasonable amount of time has passed, provided it takes care not to cause the problem to recur ("reasonable" is intentionally not defined here, since it depends on the application, media codec, and congestion control algorithm).

The RTCP reporting interval of the media sender does not affect how quickly congestion circuit breaker can trigger. The timing is based on the RTCP reporting interval of the receiver that generates the SR/RR packets from which the loss rate and RTT estimate are derived (note that RTCP requires all participants in a session to have similar reporting intervals, else the participant timeout rules in [RFC3550] will not work, so this interval is likely similar to that of the sender). If the incoming RTCP SR or RR packets are using a reduced minimum RTCP reporting interval (as specified in Section 6.2 of RFC 3550 [RFC3550] or the RTP/AVPF profile [RFC4585]), then that reduced RTCP reporting interval is used when determining if the circuit breaker is triggered.

If there are more media streams that can be reported in a single RTCP SR or RR packet, or if the size of a complete RTCP SR or RR packet exceeds the network MTU, then the receiver will report on a subset of sources in each reporting interval, with the subsets selected round-robin across multiple intervals so that all sources are eventually reported [RFC3550]. When generating such round-robin RTCP reports,

priority SHOULD be given to reports on sources that have high packet loss rates, to ensure that senders are aware of network congestion they are causing (this is an update to [RFC3550]).

4.4. RTP/AVP Circuit Breaker #4: Media Usability

Applications that use RTP are generally tolerant to some amount of packet loss. How much packet loss can be tolerated will depend on the application, media codec, and the amount of error correction and packet loss concealment that is applied. There is an upper bound on the amount of loss that can be corrected, however, beyond which the media becomes unusable. Similarly, many applications have some upper bound on the media capture to play-out latency that can be tolerated before the application becomes unusable. The latency bound will depend on the application, but typical values can range from the order of a few hundred milliseconds for voice telephony and interactive conferencing applications, up to several seconds for some video-on-demand systems.

As a final circuit breaker, RTP senders SHOULD monitor the reported packet loss and delay to estimate whether the media is likely to be suitable for the intended purpose. If the packet loss rate and/or latency is such that the media has become unusable, and has remained unusable for a significant time period, then the application SHOULD cease transmission. Similarly, receivers SHOULD monitor the quality of the media they receive, and if the quality is unusable for a significant time period, they SHOULD terminate the session. This memo intentionally does not define a bound on the packet loss rate or latency that will result in unusable media, as these are highly application dependent. Similarly, the time period that is considered significant is application dependent, but is likely on the order of seconds, or tens of seconds.

Sending media that suffers from such high packet loss or latency that it is unusable at the receiver is both wasteful of resources, and of no benefit to the user of the application. It also is highly likely to be congesting the network, and disrupting other applications. As such, the congestion circuit breaker will almost certainly trigger to stop flows where the media would be unusable due to high packet loss or latency. However, in pathological scenarios where the congestion circuit breaker does not stop the flow, it is desirable to prevent the application sending unnecessary traffic that might disrupt other uses of the network. The role of the media usability circuit breaker is to protect the network in such cases.

4.5. Ceasing Transmission

What it means to cease transmission depends on the application. This could mean stopping a single RTP flow, or it could mean that multiple bundled RTP flows are stopped. The intention is that the application will stop sending RTP data packets on a particular 5-tuple (transport protocol, source and destination ports, source and destination IP addresses), until whatever network problem that triggered the RTP circuit breaker has dissipated. RTP flows halted by the circuit breaker SHOULD NOT be restarted automatically unless the sender has received information that the congestion has dissipated, or can reasonably be expected to have dissipated. What could trigger this expectation is necessarily application dependent, but could be, for example, an indication that a competing flow has finished and freed up some capacity, or for an application running on a mobile device, that the device moved to a new location so the flow would traverse a different path if it were restarted. Ideally, a human user will be involved in the decision to try to restart the flow, since that user will eventually give up if the flows repeatedly trigger the circuit breaker. This will help avoid problems with automatic redial systems from congesting the network.

It is recognised that the RTP implementation in some systems might not be able to determine if a flow set-up request was initiated by a human user, or automatically by some scripted higher-level component of the system. These implementations MUST rate limit attempts to restart a flow on the same 5-tuple as used by a flow that triggered the circuit breaker, so that the reaction to a triggered circuit breaker lasts for at least the triggering interval [I-D.ietf-tsvwg-circuit-breaker].

The RTP circuit breaker will only trigger, and cease transmission, for media flows subject to long-term persistent congestion. Such flows are likely to have poor quality and usability for some time before the circuit breaker triggers. Implementations can monitor RTCP Reception Report blocks being returned for their media flows, and might find it beneficial to use this information to provide a user interface cue that problems are occurring, in advance of the circuit breaker triggering.

5. RTP Circuit Breakers and the RTP/AVPF and RTP/SAVPF Profiles

Use of the Extended RTP Profile for RTCP-based Feedback (RTP/AVPF) [RFC4585] allows receivers to send early RTCP reports in some cases, to inform the sender about particular events in the media stream. There are several use cases for such early RTCP reports, including providing rapid feedback to a sender about the onset of congestion. The RTP/SAVPF Profile [RFC5124] is a secure variant of the RTP/AVPF

profile, that is treated the same in the context of the RTP circuit breaker. These feedback profiles are often used with non-compound RTCP reports [RFC5506] to reduce the reporting overhead.

Receiving rapid feedback about congestion events potentially allows congestion control algorithms to be more responsive, and to better adapt the media transmission to the limitations of the network. It is expected that many RTP congestion control algorithms will adopt the RTP/AVPF profile or the RTP/SAVPF profile for this reason, defining new transport layer feedback reports that suit their requirements. Since these reports are not yet defined, and likely very specific to the details of the congestion control algorithm chosen, they cannot be used as part of the generic RTP circuit breaker.

Reduced-size RTCP reports sent under the RTP/AVPF early feedback rules that do not contain an RTCP SR or RR packet MUST be ignored by the congestion circuit breaker (they do not contain the information needed by the congestion circuit breaker algorithm), but MUST be counted as received packets for the RTCP timeout circuit breaker. Reduced-size RTCP reports sent under the RTP/AVPF early feedback rules that contain RTCP SR or RR packets MUST be processed by the congestion circuit breaker as if they were sent as regular RTCP reports, and counted towards the circuit breaker conditions specified in Section 4 of this memo. This will potentially make the RTP circuit breaker trigger earlier than it would if the RTP/AVPF profile was not used.

When using ECN with RTP (see Section 7), early RTCP feedback packets can contain ECN feedback reports. The count of ECN-CE marked packets contained in those ECN feedback reports is counted towards the number of lost packets reported if the ECN Feedback Report is sent in a compound RTCP packet along with an RTCP SR/RR report packet. Reports of ECN-CE packets sent as reduced-size RTCP ECN feedback packets without an RTCP SR/RR packet MUST be ignored.

These rules are intended to allow the use of low-overhead RTP/AVPF feedback for generic NACK messages without triggering the RTP circuit breaker. This is expected to make such feedback suitable for RTP congestion control algorithms that need to quickly report loss events in between regular RTCP reports. The reaction to reduced-size RTCP SR/RR packets is to allow such algorithms to send feedback that can trigger the circuit breaker, when desired.

The RTP/AVPF and RTP/SAVPF profiles include the T_rr_interval parameter that can be used to adjust the regular RTCP reporting interval. The use of the T_rr_interval parameter changes the behaviour of the RTP circuit breaker, as described in Section 4.

6. Impact of RTCP Extended Reports (XR)

RTCP Extended Report (XR) blocks provide additional reception quality metrics, but do not change the RTCP timing rules. Some of the RTCP XR blocks provide information that might be useful for congestion control purposes, others provide non-congestion-related metrics. With the exception of RTCP XR ECN Summary Reports (see Section 7), the presence of RTCP XR blocks in a compound RTCP packet does not affect the RTP circuit breaker algorithm. For consistency and ease of implementation, only the reception report blocks contained in RTCP SR packets, RTCP RR packets, or RTCP XR ECN Summary Report packets, are used by the RTP circuit breaker algorithm.

7. Impact of Explicit Congestion Notification (ECN)

The use of ECN for RTP flows does not affect the RTCP timeout circuit breaker (Section 4.1) or the media timeout circuit breaker (Section 4.2), since these are both connectivity checks that simply determinate if any packets are being received.

At the time of this writing, there’s no consensus on how the receipt of ECN feedback will impact the congestion circuit breaker (Section 4.3) or indeed whether the congestion circuit breaker ought to take ECN feedback into account. A future version of this memo is expected to provide guidance for implementers.

For the media usability circuit breaker (Section 4.4), ECN-CE marked packets arrive at the receiver, and if they arrive in time, they will be decoded and rendered as normal. Accordingly, receipt of such packets ought not affect the usability of the media, and the arrival of RTCP feedback indicating their receipt is not expected to impact the operation of the media usability circuit breaker.

8. Impact of Bundled Media and Layered Coding

The RTP circuit breaker operates on a per-RTP session basis. An RTP sender that participates in several RTP sessions MUST treat each RTP session independently with regards to the RTP circuit breaker.

An RTP sender can generate several media streams within a single RTP session, with each stream using a different SSRC. This can happen if bundled media are in use, when using simulcast, or when using layered media coding. By default, each SSRC will be treated independently by the RTP circuit breaker. However, the sender MAY choose to treat the flows (or a subset thereof) as a group, such that a circuit breaker trigger for one flow applies to the group of flows as a whole, and either causes the entire group to cease transmission, or the sending rate of the group to reduce by a factor of ten, depending on the RTP

circuit breaker triggered. Grouping flows in this way is expected to be especially useful for layered flows sent using multiple SSRCs, as it allows the layered flow to react as a whole, ceasing transmission on the enhancement layers first to reduce sending rate if necessary, rather than treating each layer independently. Care needs to be taken if the different media streams sent on a single transport layer flow use different DSCP values [RFC7657], [I-D.ietf-tsvwg-rtcweb-qos], since congestion could be experienced differently depending on the DSCP marking. Accordingly, RTP media streams with different DSCP values SHOULD NOT be considered as a group when evaluating the RTP Circuit Breaker conditions.

9. Security Considerations

The security considerations of [RFC3550] apply.

If the RTP/AVPF profile is used to provide rapid RTCP feedback, the security considerations of [RFC4585] apply. If ECN feedback for RTP over UDP/IP is used, the security considerations of [RFC6679] apply.

If non-authenticated RTCP reports are used, an on-path attacker can trivially generate fake RTCP packets that indicate high packet loss rates, causing the circuit breaker to trigger and disrupt an RTP session. This is somewhat more difficult for an off-path attacker, due to the need to guess the randomly chosen RTP SSRC value and the RTP sequence number. This attack can be avoided if RTCP packets are authenticated; authentication options are discussed in [RFC7201].

Timely operation of the RTP circuit breaker depends on the choice of RTCP reporting interval. If the receiver has a reporting interval that is overly long, then the responsiveness of the circuit breaker decreases. In the limit, the RTP circuit breaker can be disabled for all practical purposes by configuring an RTCP reporting interval that is many minutes duration. This issue is not specific to the circuit breaker: long RTCP reporting intervals also prevent reception quality reports, feedback messages, codec control messages, etc., from being used. Implementations are expected to impose an upper limit on the RTCP reporting interval they are willing to negotiate (based on the session bandwidth and RTCP bandwidth fraction) when using the RTP circuit breaker, as discussed in Section 4.3.

10. IANA Considerations

There are no actions for IANA.

11. Acknowledgements

The authors would like to thank Bernard Aboba, Harald Alvestrand, Ben Campbell, Alissa Cooper, Spencer Dawkins, Gorry Fairhurst, Stephen Farrell, Nazila Fough, Kevin Gross, Cullen Jennings, Randell Jesup, Mirja Kuehlewind, Jonathan Lennox, Matt Mathis, Stephen McQuistin, Simon Perreault, Eric Rescorla, Abheek Saha, Meral Shirazipour, Fabio Verdicchio, and Magnus Westerlund for their valuable feedback.

12. References

12.1. 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>.

[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>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <http://www.rfc-editor.org/info/rfc3611>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[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>.

[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>.

12.2. Informative References

[Floyd] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", Proceedings of the ACM SIGCOMM conference, 2000, DOI 10.1145/347059.347397, August 2000.

[I-D.ietf-avtcore-rtp-multi-stream] Lennox, J., Westerlund, M., Wu, Q., and D. Perkins, "Sending Multiple RTP Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-11 (work in progress), December 2015.

[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-rtcweb-qos] Jones, P., Dhesikan, S., Jennings, C., and D. Druta, "DSCP Packet Markings for WebRTC QoS", draft-ietf-tsvwg-rtcweb-qos-17 (work in progress), May 2016.

[Mathis] Mathis, M., Semke, J., Mahdavi, J., and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm", ACM SIGCOMM Computer Communication Review 27(3), DOI 10.1145/263932.264023, July 1997.

[Padhye] Padhye, J., Firoiu, V., Towsley, D., and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", Proceedings of the ACM SIGCOMM conference, 1998, DOI 10.1145/285237.285291, August 1998.

[RFC2862] Civanlar, M. and G. Cash, "RTP Payload Format for Real-Time Pointers", RFC 2862, DOI 10.17487/RFC2862, June 2000, <http://www.rfc-editor.org/info/rfc2862>.

[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>.

[RFC4103] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, DOI 10.17487/RFC4103, June 2005, <http://www.rfc-editor.org/info/rfc4103>.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <http://www.rfc-editor.org/info/rfc5104>.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <http://www.rfc-editor.org/info/rfc5124>.

[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>.

[RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, DOI 10.17487/RFC5450, March 2009, <http://www.rfc-editor.org/info/rfc5450>.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <http://www.rfc-editor.org/info/rfc5506>.

[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>.

[RFC6798] Clark, A. and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Packet Delay Variation Metric Reporting", RFC 6798, DOI 10.17487/RFC6798, November 2012, <http://www.rfc-editor.org/info/rfc6798>.

[RFC6843] Clark, A., Gross, K., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay Metric Reporting", RFC 6843, DOI 10.17487/RFC6843, January 2013, <http://www.rfc-editor.org/info/rfc6843>.

[RFC6958] Clark, A., Zhang, S., Zhao, J., and Q. Wu, Ed., "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Loss Metric Reporting", RFC 6958, DOI 10.17487/RFC6958, May 2013, <http://www.rfc-editor.org/info/rfc6958>.

[RFC7002] Clark, A., Zorn, G., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Discard Count Metric Reporting", RFC 7002, DOI 10.17487/RFC7002, September 2013, <http://www.rfc-editor.org/info/rfc7002>.

[RFC7003] Clark, A., Huang, R., and Q. Wu, Ed., "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Discard Metric Reporting", RFC 7003, DOI 10.17487/RFC7003, September 2013, <http://www.rfc-editor.org/info/rfc7003>.

[RFC7097] Ott, J., Singh, V., Ed., and I. Curcio, "RTP Control Protocol (RTCP) Extended Report (XR) for RLE of Discarded Packets", RFC 7097, DOI 10.17487/RFC7097, January 2014, <http://www.rfc-editor.org/info/rfc7097>.

[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>.

[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>.

[RFC7713] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements", RFC 7713, DOI 10.17487/RFC7713, December 2015, <http://www.rfc-editor.org/info/rfc7713>.

[Sarker] Sarker, Z., Singh, V., and C. Perkins, "An Evaluation of RTP Circuit Breaker Performance on LTE Networks", Proceedings of the IEEE Infocom workshop on Communication and Networking Techniques for Contemporary Video, 2014, April 2014.

[Singh] Singh, V., McQuistin, S., Ellis, M., and C. Perkins, "Circuit Breakers for Multimedia Congestion Control", Proceedings of the International Packet Video Workshop, 2013, DOI 10.1109/PV.2013.6691439, December 2013.

Authors' Addresses

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csperkins.org

Varun Singh
Nemu Dialogue Systems Oy
Runeberginkatu 4c A 4
Helsinki 00100
Finland

Email: varun.singh@iki.fi
URI: http://www.callstats.io/

Internet Engineering Task Force
INTERNET-DRAFT
File: draft-ietf-tcpm-rto-consider-17.txt
Intended Status: Best Current Practice
Expires: January 27, 2021

M. Allman
ICSI

July 27, 2020

Requirements for Time-Based Loss Detection

Status of this Memo

Copyright Notice

Abstract

Many protocols must detect packet loss for various reasons (e.g., to ensure reliability using retransmissions or to understand the level of congestion along a network path). While many mechanisms have been designed to detect loss, ultimately, protocols can only count on the passage of time without delivery confirmation to declare a packet "lost". Each implementation of a time-based loss detection mechanism represents a balance between correctness and timeliness and therefore no implementation suits all situations. This document

provides high-level requirements for time-based loss detectors appropriate for general use in unicast communication across the Internet. Within the requirements, implementations have latitude to define particulars that best address each situation.

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.

1 Introduction

As a network of networks, the Internet consists of a large variety of links and systems that support a wide variety of tasks and workloads. The service provided by the network varies from best-effort delivery among loosely connected components to highly predictable delivery within controlled environments (e.g., between physically connected nodes, within a tightly controlled data center). Each path through the network has a set of path properties---e.g., available capacity, delay, packet loss. Given the range of networks that make up the Internet, these properties range from largely static to highly dynamic.

This document provides guidelines for developing an understanding of one path property: packet loss. In particular, we offer guidelines for developing and implementing time-based loss detectors that have been gradually learned over the last several decades. We focus on the general case where the loss properties of a path are (a) unknown a priori and (b) dynamically vary over time. Further, while there are numerous root causes of packet loss, we leverage the conservative notion that loss is an implicit indication of congestion [RFC5681]. While this stance is not always correct, as a general assumption it has historically served us well [Jac88]. As we discuss further in section 2, the guidelines in this document should be viewed as a general default for unicast communication across best-effort networks and not as optimal---or even applicable---for all situations.

Given that packet loss is routine in best-effort networks, loss detection is a crucial activity for many protocols and applications and is generally undertaken for two major reasons:

(1) Ensuring reliable data delivery.

This requires a data sender to develop an understanding of which transmitted packets have not arrived at the receiver. This knowledge allows the sender to retransmit missing data.

(2) Congestion control.

As we mention above, packet loss is often taken as an

implicit indication that the sender is transmitting too fast and is overwhelming some portion of the network path. Data senders can therefore use loss to trigger transmission rate reductions.

Various mechanisms are used to detect losses in a packet stream. Often we use continuous or periodic acknowledgments from the recipient to inform the sender's notion of which pieces of data are missing. However, despite our best intentions and most robust mechanisms we cannot place ultimate faith in receiving such acknowledgments, but can only truly depend on the passage of time. Therefore, our ultimate backstop to ensuring that we detect all loss is a timeout. That is, the sender sets some expectation for how long to wait for confirmation of delivery for a given piece of data. When this time period passes without delivery confirmation the sender concludes the data was lost in transit.

The specifics of time-based loss detection schemes represent a tradeoff between correctness and responsiveness. In other words we wish to simultaneously:

- wait long enough to ensure the detection of loss is correct, and

- minimize the amount of delay we impose on applications (before repairing loss) and the network (before we reduce the congestion).

Serving both of these goals is difficult as they pull in opposite directions [AP99]. By not waiting long enough to accurately determine a packet has been lost we may provide a needed retransmission in a timely manner, but risk sending unnecessary ("spurious") retransmissions and needlessly lowering the transmission rate. By waiting long enough that we are unambiguously certain a packet has been lost we cannot repair losses in a timely manner and we risk prolonging network congestion.

Many protocols and applications---such as TCP [RFC6298], SCTP [RFC4960], SIP [RFC3261]---use their own time-based loss detection mechanisms. At this point, our experience leads to a recognition that often specific tweaks that deviate from standardized time-based loss detectors do not materially impact network safety with respect to congestion control [AP99]. Therefore, in this document we outline a set of high-level protocol-agnostic requirements for time-based loss detection. The intent is to provide a safe foundation on which implementations have the flexibility to instantiate mechanisms that best realize their specific goals.

2 Context

This document is different from the way we ideally like to engineer systems. Usually, we strive to understand high-level requirements as a starting point. We then methodically engineer specific protocols, algorithms and systems that meet these requirements. Within the IETF standards process we have derived many time-based

loss detection schemes without benefit from some over-arching requirements document---because we had no idea how to write such a document! Therefore, we made the best specific decisions we could in response to specific needs.

At this point, however, the community's experience has matured to the point where we can define a set of general, high-level requirements for time-based loss detection schemes. We now understand how to separate the strategies these mechanisms use that are crucial for network safety from those small details that do not materially impact network safety. The requirements in this document may not be appropriate in all cases. In particular, the guidelines in section 4 are concerned with the general case, but specific situations may allow for more flexibility in terms of loss detection because specific facets of the environment are known (e.g., when operating over a single physical link or within a tightly controlled data center). Therefore, variants, deviations or wholly different time-based loss detectors may be necessary or useful in some cases. The correct way to view this document is as the default case and not as a one-size-fits-all that is optimal in all cases.

Adding a requirements umbrella to a body of existing specifications is inherently messy and we run the risk of creating inconsistencies with both past and future mechanisms. Therefore, we make the following statements about the relationship of this document to past and future specifications:

- This document does not update or obsolete any existing RFC. These previous specifications---while generally consistent with the requirements in this document---reflect community consensus and this document does not change that consensus.

- The requirements in this document are meant to provide for network safety and, as such, SHOULD be used by all future time-based loss detection mechanisms.

- The requirements in this document may not be appropriate in all cases and, therefore, deviations and variants may be necessary in the future (hence the "SHOULD" in the last bullet). However, inconsistencies MUST be (a) explained and (b) gather consensus.

3 Scope

The principles we outline in this document are protocol-agnostic and widely applicable. We make the following scope statements about the application of the requirements discussed in Section 4:

(S.1) While there are a bevy of uses for timers in protocols---from rate-based pacing to connection failure detection and beyond---this document is focused only on loss detection.

(S.2) The requirements for time-based loss detection mechanisms in this document are for the primary or "last resort" loss detection mechanism whether the mechanism is the sole loss

repair strategy or works in concert with other mechanisms.

While a straightforward time-based loss detector is sufficient for simple protocols like DNS [RFC1034,RFC1035], more complex protocols often use more advanced loss detectors to aid performance. For instance, TCP and SCTP have methods to detect (and repair) loss based on explicit endpoint state sharing [RFC2018,RFC4960,RFC6675]. Such mechanisms often provide more timely and precise loss detection than time-based loss detectors. However, these mechanisms do not obviate the need for a "retransmission timeout" or "RTO" because---as we discuss in Section 1---only the passage of time can ultimately be relied upon to detect loss. In other words, ultimately we cannot count on acknowledgments to arrive at the data sender to indicate which packets never arrived at the receiver. In cases such as these we need a time-based loss detector to functions as a "last resort".

Also, note, that some recent proposals have incorporated time as a component of advanced loss detection methods---either as an aggressive first loss detector in certain situations or in conjunction with endpoint state sharing [DCCM13,CCDJ20,IS20]. While these mechanisms can aid timely loss recovery, the protocol ultimately leans on another more conservative timer to ensure reliability when these mechanisms break down. The requirements in this document are only directly applicable to last resort loss detection. However, we expect that many of the requirements can serve as useful guidelines for more aggressive non-last resort timers, as well.

(S.3) The requirements in this document apply only to endpoint-to-endpoint unicast communication. Reliable multicast (e.g., [RFC5740]) protocols are explicitly outside the scope of this document.

Protocols such as SCTP [RFC4960] and MP-TCP [RFC6182] that communicate in a unicast fashion with multiple specific endpoints can leverage the requirements in this document provided they track state and follow the requirements for each endpoint independently. I.e., if host A communicates with addresses B and C, A needs to use independent time-based loss detector instances for traffic sent to B and C.

(S.4) There are cases where state is shared across connections or flows (e.g., [RFC2140], [RFC3124]). State pertaining to time-based loss detection is often discussed as sharable. These situations raise issues that the simple flow-oriented time-based loss detection mechanism discussed in this document does not consider (e.g., how long to preserve state between connections). Therefore, while the general principles given in Section 4 are likely applicable, sharing time-based loss detection information across flows is outside the scope of this document.

4 Requirements

We now list the requirements that apply when designing primary or last resort time-based loss detection mechanisms. For historical reasons and ease of exposition, we refer to the time between sending a packet and determining the packet has been lost due to lack of delivery confirmation as the "retransmission timeout" or "RTO". After the RTO passes without delivery confirmation, the sender may safely assume the packet is lost. However, as discussed above, the detected loss need not be repaired (i.e., the loss could be detected only for congestion control and not reliability purposes).

(1) As we note above, loss detection happens when a sender does not receive delivery confirmation within some expected period of time. In the absence of any knowledge about the latency of a path, the initial RTO MUST be conservatively set to no less than 1 second.

Correctness is of the utmost importance when transmitting into a network with unknown properties because:

- Premature loss detection can trigger spurious retransmits that could cause issues when a network is already congested.

- Premature loss detection can needlessly cause congestion control to dramatically lower the sender's allowed transmission rate---especially since the rate is already likely low at this stage of the communication. Recovering from such a rate change can taken a relatively long time.

- Finally, as discussed below, sometimes using time-based loss detection and retransmissions can cause ambiguities in assessing the latency of a network path. Therefore, it is especially important for the first latency sample to be free of ambiguities such that there is a baseline for the remainder of the communication.

The specific constant (1 second) comes from the analysis of Internet RTTs found in Appendix A of [RFC6298].

(2) We now specify four requirements that pertain to setting an expected time interval for delivery confirmation.

Often measuring the time required for delivery confirmation is framed as assessing the "round-trip time (RTT)" of the network path. The RTT is the minimum amount of time required to receive delivery confirmation and also often follows protocol behavior whereby acknowledgments are generated quickly after data arrives. For instance, this is the case for the RTO used by TCP [RFC6298] and SCTP [RFC4960]. However, this is somewhat mis-leading and the expected latency is better framed as the "feedback time" (FT). In other words, the expectation is not always simply a network property, but can include additional time before a sender should reasonably expect a response.

For instance, consider a UDP-based DNS request from a client to a recursive resolver [RFC1035]. When the request can be served from the resolver's cache the FT likely well approximates the network RTT between the client and resolver. However, on a cache miss the resolver will request the needed information from one or more authoritative DNS servers, which will non-trivially increase the FT compared to the network RTT between the client and resolver.

Therefore, we express the requirements in terms of FT. Again, for ease of exposition we use "RTO" to indicate the interval between a packet transmission and the decision the packet has been lost---regardless of whether the packet will be retransmitted.

(a) The RTO SHOULD be set based on multiple observations of the FT when available.

    In other words, the RTO should represent an empirically-derived reasonable amount of time that the sender should wait for delivery confirmation before deciding the given data is lost. Network paths are inherently dynamic and therefore it is crucial to incorporate multiple recent FT samples in the RTO to take into account the delay variation across time.

    For example, TCP's RTO [RFC6298] would satisfy this requirement due to its use of an exponentially-weighted moving average (EWMA) to combine multiple FT samples into a "smoothed RTT". In the name of conservativeness, TCP goes further to also include an explicit variance term when computing the RTO.

    While multiple FT samples are crucial for capturing the delay dynamics of a path, we explicitly do not tightly specify the process---including the number of FT samples to use and how/when to age samples out of the RTO calculation---as the particulars could depend on the situation and/or goals of each specific loss detector.

    Finally, FT samples come from packet exchanges between peers. We encourage protocol designers---especially for new protocols---to strive to ensure the feedback is not easily spoofable by on- or off-path attackers such that they can perturb a host's notion of the FT. Ideally, all messages would be cryptographically secure, but given that this is not always possible---especially in legacy protocols---using a healthy amount of randomness in the packets is encouraged.

(b) FT observations SHOULD be taken and incorporated into the RTO at least once per RTT or as frequently as data is exchanged in cases where that happens less frequently than once per RTT.

Internet measurements show that taking only a single FT sample per TCP connection results in a relatively poorly performing RTO mechanism [AP99], hence this requirement that the FT be sampled continuously throughout the lifetime of communication.

As an example, TCP takes an FT sample roughly once per RTT, or if using the timestamp option [RFC7323] on each acknowledgment arrival. [AP99] shows that both these approaches result in roughly equivalent performance for the RTO estimator.

(c) FT observations MAY be taken from non-data exchanges.

Some protocols use non-data exchanges for various reasons---e.g., keepalives, heartbeats, control messages. To the extent that the latency of these exchanges mirrors data exchange, they can be leveraged to take FT samples within the RTO mechanism. Such samples can help protocols keep their RTO accurate during lulls in data transmission. However, given that these messages may not be subject to the same delays as data transmission, we do not take a general view on whether this is useful or not.

(d) An RTO mechanism MUST NOT use ambiguous FT samples.

Assume two copies of some packet X are transmitted at times t0 and t1 and then at time t2 the sender receives confirmation that X in fact arrived. In some cases, it is not clear which copy of X triggered the confirmation and hence the actual FT is either t2-t1 or t2-t0, but which is a mystery. Therefore, in this situation an implementation MUST NOT use either version of the FT sample and hence not update the RTO (as discussed in [KP87,RFC6298]).

There are cases where two copies of some data are transmitted in a way whereby the sender can tell which is being acknowledged by an incoming ACK. E.g., TCP's timestamp option [RFC7323] allows for packets to be uniquely identified and hence avoid the ambiguity. In such cases there is no ambiguity and the resulting samples can update the RTO.

(3) Loss detected by the RTO mechanism MUST be taken as an indication of network congestion and the sending rate adapted using a standard mechanism (e.g., TCP collapses the congestion window to one packet [RFC5681]).

This ensures network safety.

An exception to this rule is if an IETF standardized mechanism determines that a particular loss is due to a non-congestion event (e.g., packet corruption). In such a case a congestion

control action is not required. Additionally, congestion control actions taken based on time-based loss detection could be reversed when a standard mechanism post-facto determines that the cause of the loss was not congestion (e.g., [RFC5682]).

(4) Each time the RTO is used to detect a loss, the value of the RTO MUST be exponentially backed off such that the next firing requires a longer interval. The backoff SHOULD be removed after either (a) the subsequent successful transmission of non-retransmitted data, or (b) an RTO passes without detecting additional losses. The former will generally be quicker. The latter covers cases where loss is detected, but not repaired.

A maximum value MAY be placed on the RTO. The maximum RTO MUST NOT be less than 60 seconds (as specified in [RFC6298]).

This ensures network safety.

As with guideline (3), an exception to this rule exists if an IETF standardized mechanism determines that a particular loss is not due to congestion.

5 Discussion

We note that research has shown the tension between the responsiveness and correctness of time-based loss detection seems to be a fundamental tradeoff in the context of TCP [AP99]. That is, making the RTO more aggressive (e.g., via changing TCP's exponentially weighted moving average (EWMA) gains, lowering the minimum RTO, etc.) can reduce the time required to detect actual loss. However, at the same time, such aggressiveness leads to more cases of mistakenly declaring packets lost that ultimately arrived at the receiver. Therefore, being as aggressive as the requirements given in the previous section allow in any particular situation may not be the best course of action because detecting loss---even if falsely---carries a requirement to invoke a congestion response which will ultimately reduce the transmission rate.

While the tradeoff between responsiveness and correctness seems fundamental, the tradeoff can be made less relevant if the sender can detect and recover from mistaken loss detection. Several mechanisms have been proposed for this purpose, such as Eifel [RFC3522], F-RTO [RFC5682] and DSACK [RFC2883,RFC3708]. Using such mechanisms may allow a data originator to tip towards being more responsive without incurring (as much of) the attendant costs of mistakenly declaring packets to be lost.

Also, note that, in addition to the experiments discussed in [AP99], the Linux TCP implementation has been using various non-standard RTO mechanisms for many years seemingly without large-scale problems (e.g., using different EWMA gains than specified in [RFC6298]). Further, a number of TCP implementations use a steady-state minimum RTO that is less than the 1 second specified in [RFC6298]. While the implication of these deviations from the standard may be more

spurious retransmits (per [AP99]), we are aware of no large-scale network safety issues caused by this change to the minimum RTO. This informs the guidelines in the last section (e.g., there is no minimum RTO specified).

Finally, we note that while allowing implementations to be more aggressive could in fact increase the number of needless retransmissions, the above requirements fail safe in that they insist on exponential backoff and a transmission rate reduction. Therefore, providing implementers more latitude than they have traditionally been given in IETF specifications of RTO mechanisms does not somehow open the flood gates to aggressive behavior. Since there is a downside to being aggressive, the incentives for proper behavior are retained in the mechanism.

6 Security Considerations

This document does not alter the security properties of time-based loss detection mechanisms. See [RFC6298] for a discussion of these within the context of TCP.

7 IANA Considerations

This document has no IANA considerations.

Acknowledgments

This document benefits from years of discussions with Ethan Blanton, Sally Floyd, Jana Iyengar, Shawn Ostermann, Vern Paxson, and the members of the TCPM and TCP-IMPL working groups. Ran Atkinson, Yuchung Cheng, David Black, Stewart Bryant, Martin Duke, Wesley Eddy, Gorry Fairhurst, Rahul Arvind Jadhav, Benjamin Kaduk, Mirja Kuhlewind, Nicolas Kuhn, Jonathan Looney and Michael Scharf provided useful comments on previous versions of this document.

Normative References

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017.

Informative References

[AP99] Allman, M., V. Paxson, "On Estimating End-to-End Network Path Properties", Proceedings of the ACM SIGCOMM Technical Symposium, September 1999.

[CCDJ20] Cheng, Y., N. Cardwell, N. Dukkipati, P. Jha, "RACK: a time-based fast loss detection algorithm for TCP", Internet-Draft draft-ietf-tcpm-rack-08.txt (work in progress), March 2020.

[DCCM13] Dukkipati, N., N. Cardwell, Y. Cheng, M. Mathis, "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses", Internet-Draft draft-dukkipati-tcpm-tcp-loss-probe-01.txt (work in progress), February 2013.

[IS20] Iyengar, I., I. Swett, "QUIC Loss Detection and Congestion Control", Internet-Draft draft-ietf-quic-recovery-27.txt (work in progress), March 2020.

[Jac88] Jacobson, V., "Congestion Avoidance and Control", ACM SIGCOMM, August 1988.

[KP87] Karn, P. and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols", SIGCOMM 87.

[RFC1034] Mockapetris, P. "Domain Names - Concepts and Facilities", RFC 1034, November 1987.

[RFC1035] Mockapetris, P. "Domain Names - Implementation and Specification", RFC 1035, November 1987.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[RFC2140] Touch, J., "TCP Control Block Interdependence", RFC 2140, April 1997.

[RFC2883] Floyd, S., Mahdavi, J., Mathis, M., and M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP", RFC 2883, July 2000.

[RFC3124] Balakrishnan, H., S. Seshan, "The Congestion Manager", RFC 3124, June 2001.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3522] Ludwig, R., M. Meyer, "The Eifel Detection Algorithm for TCP", RFC 3522, april 2003.

[RFC3708] Blanton, E., M. Allman, "Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions", RFC 3708, February 2004.

[RFC4960] Stweart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5681] Allman, M., V. Paxson, E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[RFC5682] Sarolahti, P., M. Kojo, K. Yamamoto, M. Hata, "Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious

Retransmission Timeouts with TCP", RFC 5682, September 2009.

[RFC5740] Adamson, B., C. Bormann, M. Handley, J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, November 2009.

[RFC6182] Ford, A., C. Raiciu, M. Handley, S. Barre, J. Iyengar, "Architectural Guidelines for Multipath TCP Development", March 2011, RFC 6182.

[RFC6298] Paxson, V., M. Allman, H.K. Chu, M. Sargent, "Computing TCP's Retransmission Timer", June 2011, RFC 6298.

[RFC6675] Blanton, E., M. Allman, L. Wang, I. Jarvinen, M. Kojo, Y. Nishida, "A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP", August 2012, RFC 6675.

[RFC7323] Borman D., B. Braden, V. Jacobson, R. Scheffenegger, "TCP Extensions for High Performance", September 2014, RFC 7323.

Authors' Addresses

Mark Allman
International Computer Science Institute
1947 Center St. Suite 600
Berkeley, CA 94704

EMail: mallman@icir.org
http://www.icir.org/mallman

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

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 October 6, 2016.

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:

- anomalous traffic that exceeds the provisioned capacity (or whose traffic characteristics exceed the threshold configured for the Circuit Breaker);

- traffic generated by an application at a time when the provisioned network capacity is being utilised for other purposes;
- routing changes that cause additional traffic to start using the path monitored by the Circuit Breaker;
- misconfiguration of a service/network device where the capacity available is insufficient to support the current traffic aggregate;
- 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:

- 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.

- Slow-Trip Circuit Breakers: This circuit breaker utilizes a longer timescale and is designed to protect network traffic from congestion by traffic aggregates.

- 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.

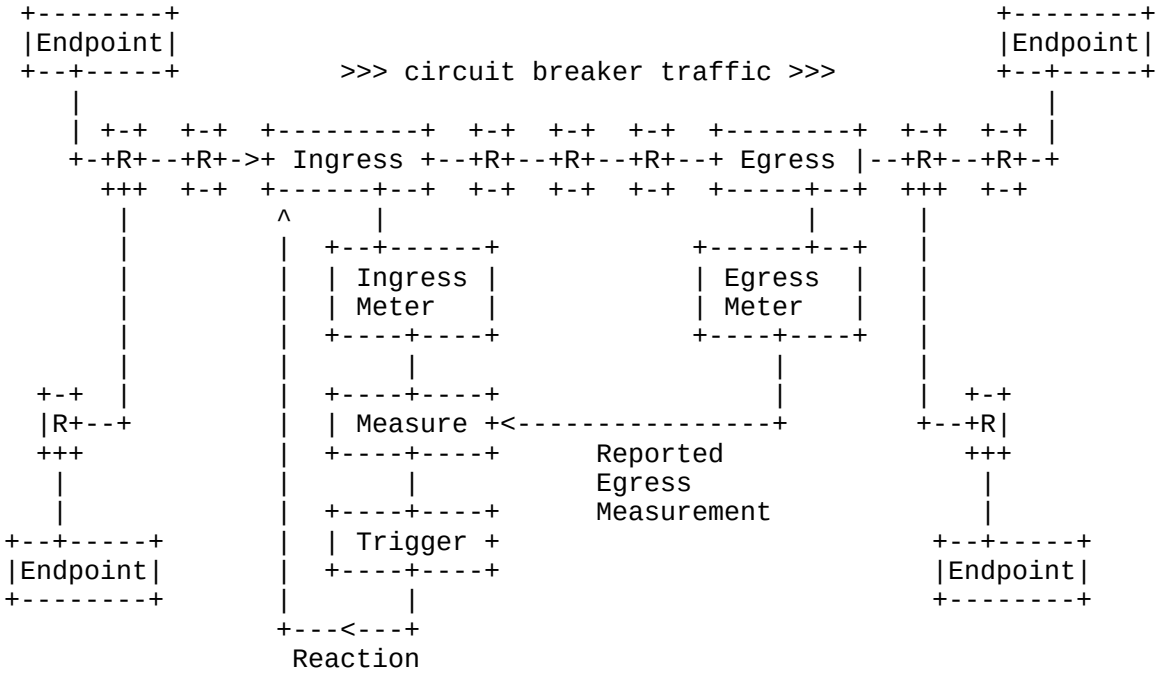


Figure 1: A CB controlling the part of the end-to-end path between an ingress point and an egress point. (Note: In some cases, the trigger

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.

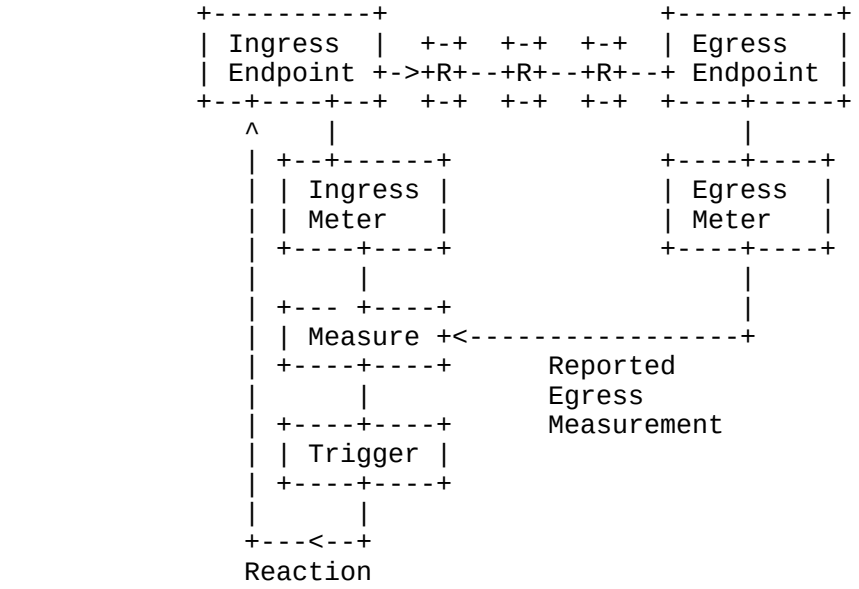


Figure 2: An endpoint CB implemented at the sender (ingress) and receiver (egress).

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

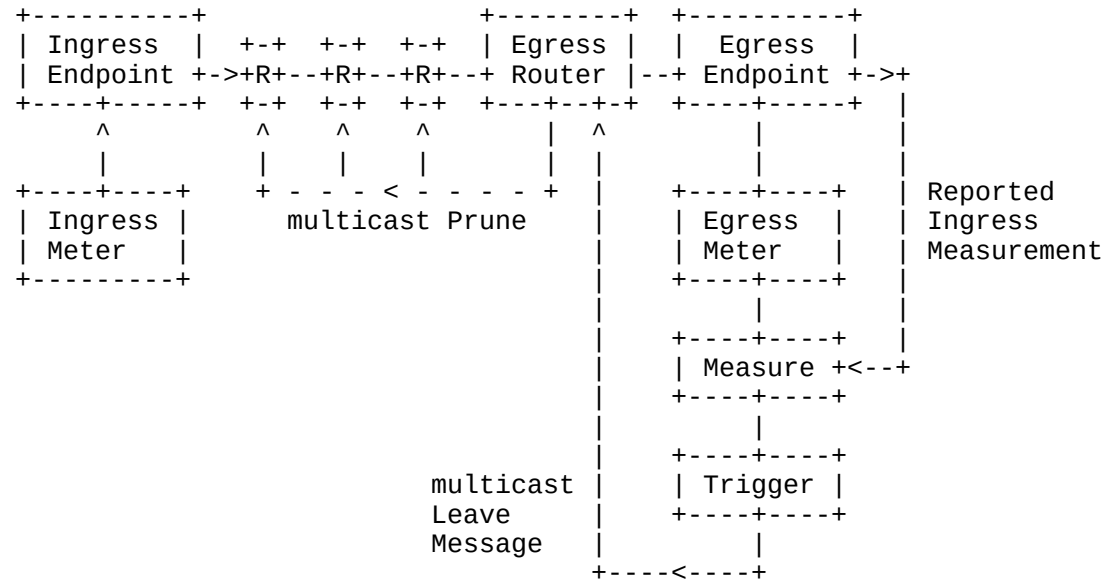


Figure 3: An example of a multicast CB controlling the end-to-end path between an ingress endpoint and an egress endpoint.

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

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)

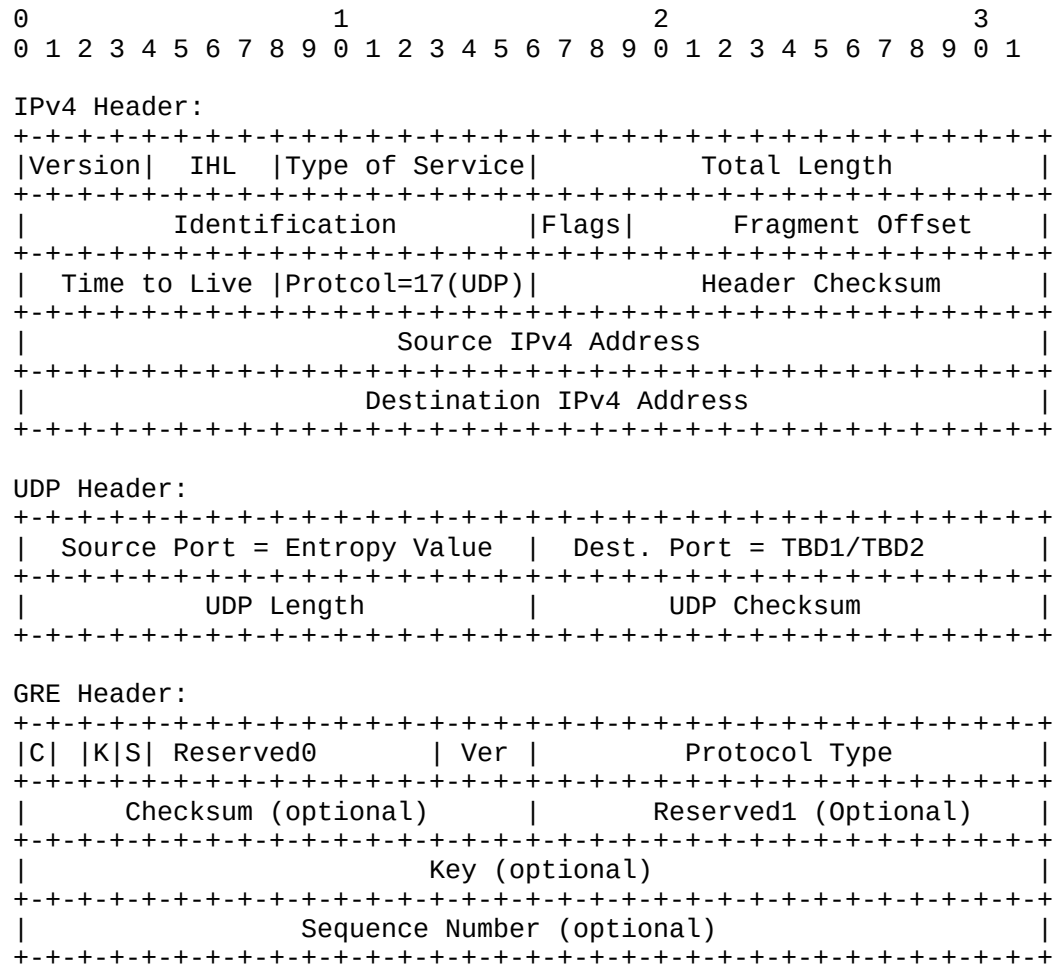


Figure 1 UDP+GRE Headers in IPv4

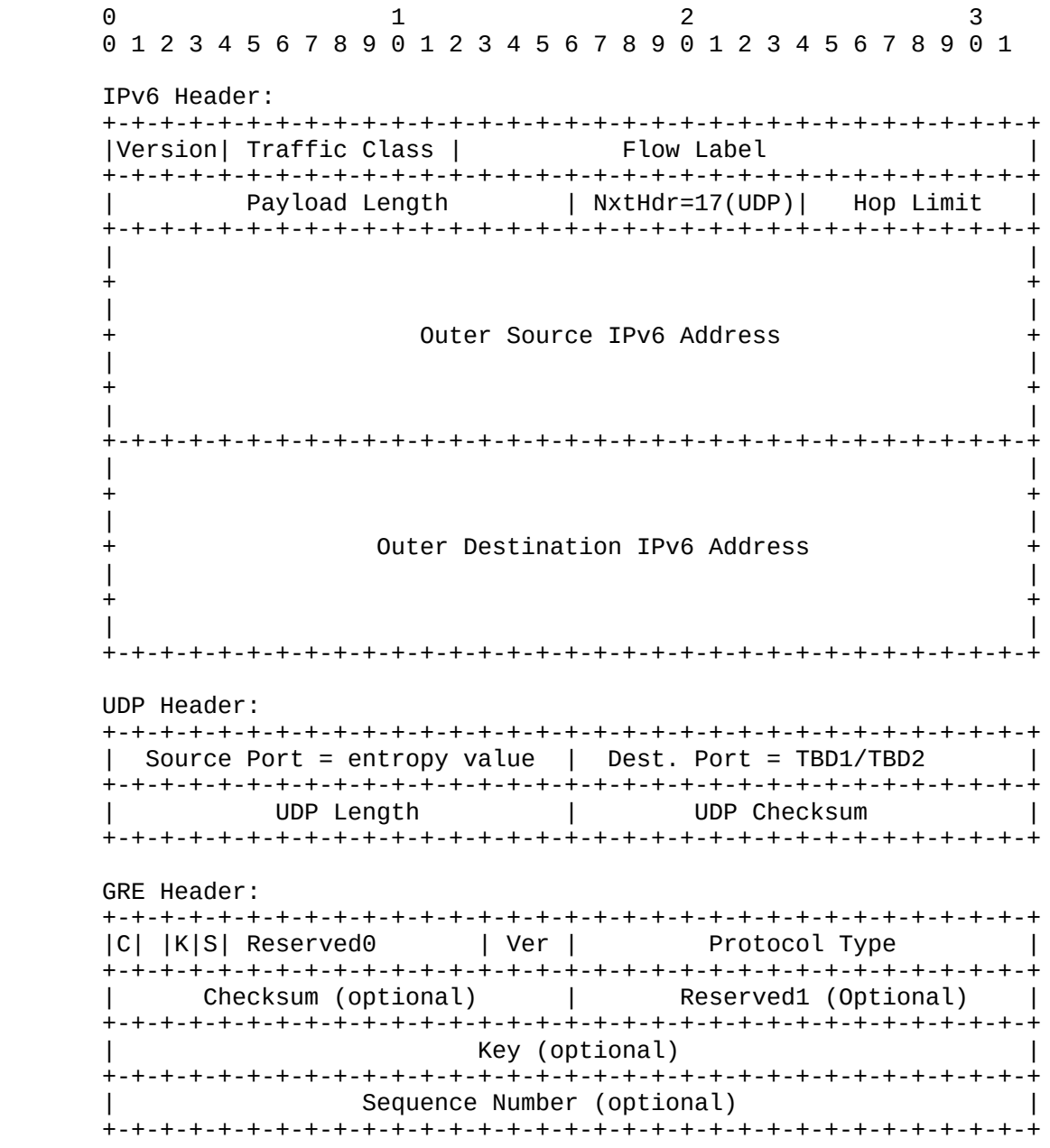


Figure 2 UDP+GRE Headers in IPv6

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.:

- Physical or logical isolation of the links carrying GRE-in-UDP from the general Internet.
- Deployment of packet filters that block the UDP ports assigned for GRE-in-UDP.
- 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

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

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/.

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:

- 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.

- 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.

- 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.)

- 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.

- 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.:

- Requirements for tunnels that carry or encapsulate using ECN code points [RFC6040].
- Usage of the IP DSCP field by tunnel endpoints [RFC2983].
- Encapsulation considerations in the design of tunnels [I-D.ietf-rtgwg-dt-encap].
- Usage of ICMP messages [I-D.ietf-intarea-tunnels].
- Handling of fragmentation and packet size for tunnels [I-D.ietf-intarea-tunnels].
- Source port usage for tunnels designed to support equal cost multipath (ECMP) routing (see Section 5.1.1).
- 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).
- 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:

- 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.

- 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].

- 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].

- 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:

- 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).

- 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.

- 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 SHOULD use a full-featured transport (e.g., TCP)	3
SHOULD control rate of transmission SHOULD perform congestion control over all traffic	3.1
for bulk transfers, SHOULD consider implementing TFRC else, SHOULD in other ways use bandwidth similar to TCP	3.1.2
for non-bulk transfers, SHOULD measure RTT and transmit max. 1 datagram/RTT else, SHOULD send at most 1 datagram every 3 seconds SHOULD back-off retransmission timers following loss	3.1.3 3.1.1
SHOULD provide mechanisms to regulate the bursts of transmission	3.1.6
MAY implement ECN; a specific set of application mechanisms are REQUIRED if ECN is used.	3.1.7
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 non-CC controlled flows SHOULD implement a transport circuit breaker MAY implement a circuit breaker for other applications	3.1.10
for tunnels carrying IP traffic, SHOULD NOT perform congestion control MUST correctly process the IP ECN field	3.1.11
for non-IP tunnels or rate not determined by traffic, SHOULD perform CC or use circuit breaker SHOULD restrict types of traffic transported by the tunnel	3.1.11
SHOULD NOT send datagrams that exceed the PMTU, i.e., SHOULD discover PMTU or send datagrams < minimum PMTU; Specific application mechanisms are REQUIRED if PLPMTUD is used.	3.2
SHOULD handle datagram loss, duplication, reordering SHOULD be robust to delivery delays up to 2 minutes	3.3

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 else, MAY use UDP-Lite with suitable checksum coverage	5.1 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:

- Addressed IESG review comments.

Changes in draft-ietf-tsvwg-rfc5405bis-18:

- Fix a nit.

Changes in draft-ietf-tsvwg-rfc5405bis-17:

- Incorporated text from Mark Allman for the section on "Protocol Timer Guidelines".

Changes in draft-ietf-tsvwg-rfc5405bis-16:

- Addressed suggestions by David Black.

Changes in draft-ietf-tsvwg-rfc5405bis-15:

- Addressed more suggestions by Takeshi Takahashi.

Changes in draft-ietf-tsvwg-rfc5405bis-14:

- Addressed SEC_DIR review by Takeshi Takahashi.

- Addressed Gen-ART review by Paul Kyzivat.

- Addressed OPS-DIR review by Tim Chown.

- Addressed some of Mark Allman’s comments regarding RTTs and RTOs.

- Addressed some of Brian Trammell;s comments regarding new IETF transports.

Changes in draft-ietf-tsvwg-rfc5405bis-13:

- Minor corrections.

- Changes recommended by Spencer Dawkins.

- Placed the recommendations on timers within section 3.1

- Updated the recommendations on reliability to also reference the recommendations on timers.

Changes in draft-ietf-tsvwg-rfc5405bis-12:

- Introduced a separate section on the usage of timers to avoid repeating similar guidance in multiple sections.

- Updated RTT measurement text to align with revised min RTO recommendation for TCP.

- Updated text based on draft-ietf-tcpm-rto-consider-03 and to now cite this draft.

- Fixed inconsistency in term used for keep-alive messages (keep-alive packet, keep-alives).

Changes in draft-ietf-tsvwg-rfc5405bis-11:

- Address some issues that idnits found.

Changes in draft-ietf-tsvwg-rfc5405bis-10:

- Restored changes from -08 that -09 accidentally rolled back.

Changes in draft-ietf-tsvwg-rfc5405bis-09:

- Fix to cross reference in summary table.

Changes in draft-ietf-tsvwg-rfc5405bis-08:

This update introduces new text in the following sections:

- 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.

- Added reference to RFC2983 on DSCP with tunnels.

- New text after comment from David Black on needing to improve the header protection text.

- Replaced replace /controlled network environment/ with /controlled environment/ to be more consistent with other drafts.

- Section 3.1.7 now explicitly refers to the applicability subsection describing controlled environments.

- PLPMTUD section updated.

- Reworded checksum text to place IPv6 UDP zero checksum text in a separate subsection (this became too long in the main section)

- Updated summary table

Changes in draft-ietf-tsvwg-rfc5405bis-07:

This update introduces new text in the following sections:

- Addressed David Black's review during WG LC.

Changes in draft-ietf-tsvwg-rfc5405bis-06:

This update introduces new text in the following sections:

- Multicast Congestion Control Guidelines (Section rewritten by Greg and Gorry to differentiate sender-driven and receiver-driven CC)
- Using UDP Ports (Added a short para on RPF checks protecting from off-path attacks)
- Applications using Multiple UDP Ports (Added text on layered multicast)

Changes in draft-ietf-tsvwg-rfc5405bis-05:

- Amended text in section discussing RTT for CC (feedback from Colin)

Changes in draft-ietf-tsvwg-rfc5405bis-04:

- Added text on consent freshness (STUN) - (From Colin)
- Reworked text on ECN (From David)
- Reworked text on RTT with CC (with help from Mirja)
- Added references to [RFC7675], [I-D.ietf-rtgwg-dt-encap], [I-D.ietf-intarea-tunnels] and [RFC7510]

Changes in draft-ietf-tsvwg-rfc5405bis-03:

- Mention crypto hash in addition to CRC for integrity protection. (From Magnus.)
- Mention PCP. (From Magnus.)
- More accurate text on secure RTP (From Magnus.)
- Reordered abstract to reflect .bis focus (Gorry)
- Added a section on ECN, with actual ECN requirements (Gorry, help from Mirja)
- Added section on Implications of RTT on Congestion Control (Gorry)

- Added note that this refers to other protocols over IP (E Nordmark, rtg encaps guidance)
- Added reordering text between sessions (consistent with use of ECMP, rtg encaps guidance)
- Reworked text on off-path data protection (port usage)
- Updated summary table

Changes in draft-ietf-tsvwg-rfc5405bis-02:

- Added note that guidance may be applicable beyond UDP to abstract (from Erik Nordmark).
- Small editorial changes to fix English nits.
- Added a circuit may provide benefit to CC tunnels by controlling envelope.
- Added tunnels should ingress-filter by packet type (from Erik Nordmark).
- Added tunnels should perform IETF ECN processing when supporting ECN.
- Multicast apps may employ CC or a circuit breaker.
- Added programming guidance on off-path attacks (with C. Perkins).
- Added reference to ECN benefits.

Changes in draft-ietf-tsvwg-rfc5405bis-01:

- Added text on DSCP-usage.
- 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.
- Added description of diffuse usage.
- 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:

- 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):

- The words "application designers" were removed from the draft title and the wording of the abstract was clarified abstract.

- 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 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

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 20, 2017.

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:

- media flows which are RTP streams [I-D.ietf-rtcweb-rtp-usage]
- 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:

- 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.

- 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: 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

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 March 5, 2018.

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.

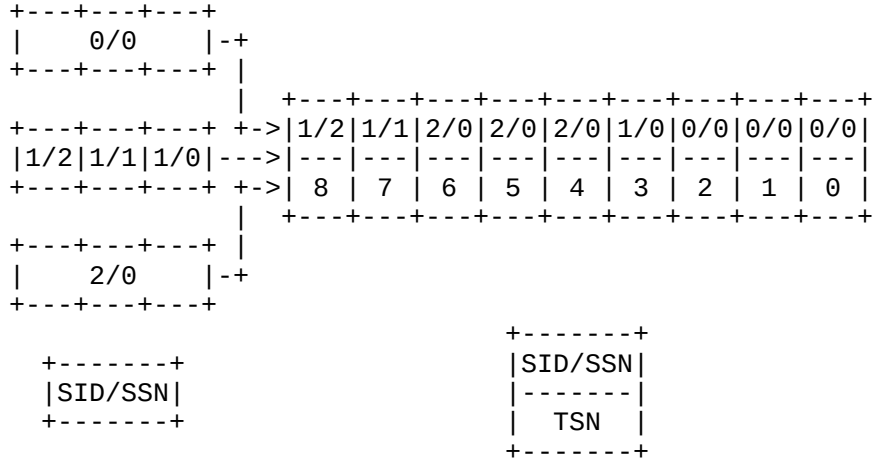


Figure 1: Round Robin Scheduler without User Message Interleaving

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.

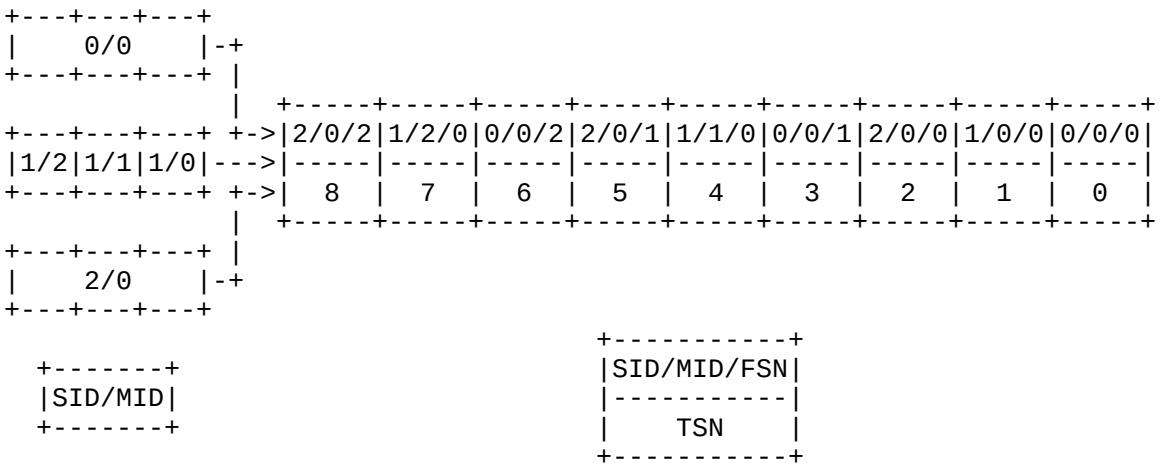


Figure 2: Round Robin Scheduler with User Message Interleaving

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.

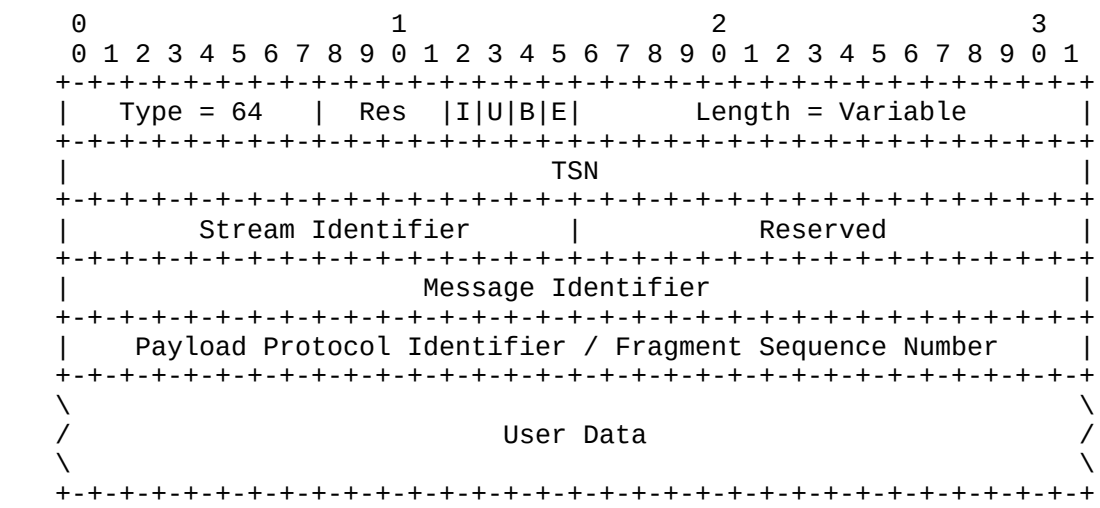


Figure 3: I-DATA chunk format

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.

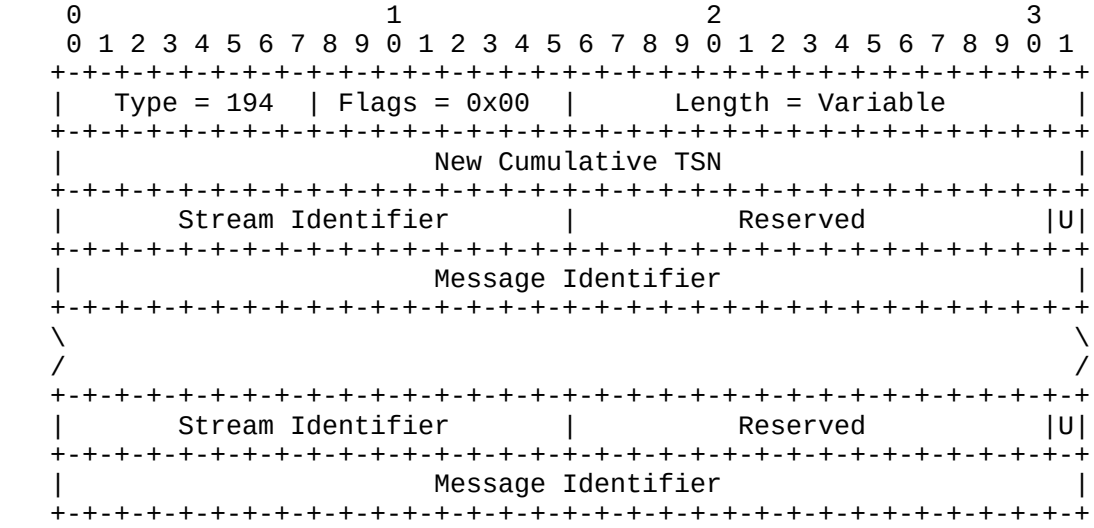


Figure 4: I-FORWARD-TSN chunk format

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

Network Working Group
Internet-Draft
Updates: 3168 (if approved)
Intended status: Experimental
Expires: October 5, 2016

N. Khademi
M. Welzl
University of Oslo
G. Armitage
Swinburne University of Technology
G. Fairhurst
University of Aberdeen

April 03, 2016

TCP Alternative Backoff with ECN (ABE)
draft-khademi-alternativebackoff-ecn-03

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). The updated method reduces cwnd by a smaller amount than TCP does in reaction 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

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 October 5, 2016.

Copyright Notice

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 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 standard TCP flow using this reaction needs significant network queue space: it can only fully utilise 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 utilise paths well even when the bottleneck queue is 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) 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 [I-D.CoDel] and PIE [I-D.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]) while allowing short traffic bursts into the queue. 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 under-utilisation 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 [RFC3168], which recommended that TCP senders respond in the same way 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:

FlightSize_(n+1) = FlightSize_n * beta_{loss}

and in response to an indication of a received ECN CE-mark:

FlightSize_(n+1) = FlightSize_n * beta_{ecn}

where, as in [RFC5681], FlightSize is the amount of outstanding data in the network, upper-bounded by the sender's congestion window (cwnd) and the receiver's advertised window (rwnd). 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 has already been running with at least two different beta_{loss} values for several years: the value in [RFC5681] is 0.5, and Linux CUBIC uses 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. NEW: 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 it 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. This indication of congestion could be treated in the same way as a congestion loss, however reception of the ECN-Echo flag SHOULD produce a reduction in FlightSize that is less than the reduction had the flow experienced loss. The reduction needs to be sufficient to allow flows sharing a bottleneck to increase their share of the capacity. This reduction MUST be less than 0.85 (at least a 15% reduction).

An ECN-capable network device cannot eliminate the possibility of 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]). Whatever the cause of loss, detection of a missing packet needs to trigger the standard loss-based congestion control response. This explicitly does not update this behaviour.

In addition, this document RECOMMENDS that experimental deployments method multiply the FlightSize by 0.8 and reduce the slow start threshold ’ssthresh’ in response to reception of a TCP segment that sets the ECN-Echo flag."

3.3. Status of the Update

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.

The present specification has been assigned an Experimental status, to provide Internet deployment experience before being proposed as a Standards-Track update.

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. Thanks to (in alphabetical order) Bob Briscoe, John Leslie, Dave Taht and the TCPM WG for providing valuable feedback on this document.

The authors would like to thank feedback on the congestion control behaviour specified in this update received from the IRTF Internet Congestion Control Research Group (ICCRG).

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 with ECN that will typically lead to a change in the capacity achieved when flows share a network bottleneck. Similar unfairness in the way that

capacity is shared 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 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-08, November 2015.

[I-D.CoDel]
Nichols, K., Jacobson, V., McGregor, V., and J. Iyengar, "The Benefits of using Explicit Congestion Notification (ECN)", Internet-draft, IETF work-in-progress draft-ietf-aqm-codel-02, December 2015.

[I-D.PIE] Pan, R., Natarajan, P., Baker, F., White, G., VerSteeg, B., Prabhu, M., Piglione, C., and V. Subramanian, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem", Internet-draft, IETF work-in-progress draft-ietf-aqm-pie-03, November 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
Updates: 3168,4774 (if approved)
Intended status: Standards Track
Expires: January 21, 2017

N. Khademi
M. Welzl
University of Oslo
G. Armitage
Swinburne University of Technology
G. Fairhurst
University of Aberdeen

July 20, 2016

Updating the Explicit Congestion Notification (ECN) Specification to Allow IETF Experimentation
draft-khademi-tsvwg-ecn-response-01

Abstract

This document relaxes recommendations and prescriptions from RFC3168 and RFC4774 that get in the way of experimentation with different ECN strategies. First, RFC3168 and RFC4774 state that, upon the receipt by an ECN-Capable transport of a single CE packet, the congestion control algorithms followed at the end-systems MUST be essentially the same as the congestion control response to a single dropped packet. This document relaxes this rule in order to encourage experimentation with different backoff strategies. Second, this document allows future IETF specifications to use the ECT(1) codepoint in ways that are currently prohibited by RFC3168. Third, this document allows future IETF experiments to use the ECT(0) or ECT(1) codepoint on any TCP segment.

Table of Contents

1. Introduction

This document relaxes three limitations that are due to specific text in [RFC3168] and, in one case, also [RFC4774].

1.1. Differently reacting to ECN-marks and loss

Explicit Congestion Notification (ECN) as specified in [RFC3168] 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, this 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. This corresponds to a backoff multiplier of 0.5 (halving cwnd and sshthresh after packet loss). Consequently, a standard TCP flow using this reaction needs significant network queue space: it can only fully utilise 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 multiplier of 0.5 is not the only available strategy. As defined in [I-D.CUBIC], CUBIC multiplies the current cwnd by 0.7 in response to loss (the Linux implementation of CUBIC has used a multiplier of 0.7 since kernel version 2.6.25 released in 2008). Consequently, CUBIC utilises paths well even when the bottleneck queue is 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 average 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. Such behavior is however prohibited

by the rules in [RFC3168].

ECN has seen little deployment so far. 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. [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. However, the limitation of [RFC3168] restricts a sender to react to notification of a CE-mark in the same way as if a packet was lost. This prohibits experimentation with ECN mechanisms that could yield greater benefits. This specification therefore relaxes this constraint.

1.1.1. Discussion: Why Use ECN to Vary the Degree of Backoff?

The classic rule-of-thumb dictates that a transport provides 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 sufficient to maintain 100% utilisation of the network path.

AQM schemes like CoDel [I-D.CoDel] and PIE [I-D.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]) while allowing short traffic bursts into the queue. This interacts acceptably for TCP connections over low BDP paths, or highly multiplexed scenarios (many concurrent TCP connections). However, it interacts badly with lightly-multiplexed cases (few concurrent connections) over a high BDP path. Conventional TCP backoff in such cases leads to gaps in packet transmission and under-utilisation of the path.

The idea to react differently to loss upon detecting an ECN CE-mark pre-dates [ABE2015]. [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 work with RED-based bottlenecks that were not necessarily configured to emulate a shallow queue.

This update to [RFC3168] that enables the IETF to specify experiments

with a different backoff behavior in response to a CE-mark than in response to packet loss is utilized by an experiment called "Alternative Backoff with ECN" (ABE). ABE is based upon [ABE2015] and defined in [I-D.ABE].

1.2. Senders setting the ECT(1) codepoint

Future IETF experiments may require setting the ECT(0) or ECT(1) codepoints differently from what [RFC3168] recommends or requires.

[NOTE: This usage was also specified in ECN-NONCE.]

This update may also allow the iETF to specify future mechanisms that associate alternate ECN semantics with this codepoint. An experiment called "L4S" proposes to use the ECT(1) codepoint to indicate in which of two queues a packet should be placed [I-D.briscoe-tsvwg-ecn-l4s-id].

1.3. ECT(0) and ECT(1) on control packets

Diverging from recommendations or requirements in [RFC3168], future IETF experiments may be specified to use the ECT(0) or ECT(1) codepoint. This choice of codepoint can be used to signal alternative ECN semantics. This supersedes the rationale in Section 20 of [RFC3168] that argued against the use of ECT(1) to specify alternate ECN semantics, instead arguing for attaching specific ECN semantics to a Differentiated Services Code Point (DSCP).

This update may also allow the iETF to specify future updates to transport protocol use of ECN. A proposal,[I-D.bagnulo-tsvwg-generalized-ecn], provides arguments for using the ECT(0) or ECT(1) codepoint on a broader range of TCP packets for which such usage is precluded by [RFC3168]: SYNs, pure ACKs, retransmitted packets and window probe packets.

2. Updating RFC3168 and RFC4774

This section specifies updates to [RFC3168] (and corresponding text in [RFC4774]) and refers to experiments that are possible within the framework provided by the update.

2.1. 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].

2.2. Scope of this update

Internet deployment of new mechanisms enabled by this update REQUIRE IETF specification in an Experimental or a Standards Track RFC approved by the IESG.

Some mechanisms rely on ECN semantics that differ from the definitions in [RFC3168] -- for example, Congestion Exposure (ConEx) [RFC7713] and DCTCP [I-D.ietf-tcpm-dctcp] need more accurate ECN information than the feedback mechanism in [RFC3168] offers (defined in [I-D.ietf-tcpm-accurate-ecn]). Such mechanisms allow a sending rate adjustment more frequent than each RTT. These mechanisms are out of the scope of the current document.

The remainder of this section lists a set of changes to [RFC3168] that are not specific replacements of text passages.

2.3. Changes to the meaning of a CE-Mark codepoint

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.

[RFC3168] and [RFC4774] contain the following text:

"Upon the receipt by an ECN-Capable transport of a single CE packet, the congestion control algorithms followed at the end-systems MUST be essentially the same as the congestion control response to a *single* dropped packet. For example, for ECN-Capable TCP the source TCP is required to halve its congestion window for any window of data containing either a packet drop or an ECN indication."

This memo updates the preceding text by replacing it with the following text:

"Upon the receipt by an ECN-Capable transport of a single CE-Marked packet, the congestion control algorithms followed at the endpoints MUST make a congestion control response as specified in [RFC3168] or its updates. For example, an ECN-Capable TCP sender could halve its congestion window for any window of data containing either a packet drop or an ECN indication."

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 the preceding text by replacing it with the following text:

"If a TCP sender receives an indication of a receibed 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. An indication of congestion, signalled by reception of the ECN-Echo flag (with the semantics defined in [RFC3168]) MUST produce a rate reduction of at least 15%, so that flows sharing the same bottleneck can increase their share of the capacity. The indication of congestion could be treated in the same way as if the flow had experienced loss, but future congestion control methods are allowed to specify a reduction that is less than the reduction for congestion loss.

An ECN-capable network device cannot eliminate the possibility of packet loss. A drop may still 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]). Whatever the cause of loss, detection of a missing packet needs to trigger the standard loss-based congestion control response". This update explicitly does not change the use of standard protocol mechanisms following loss, as required in [RFC3168].

2.4. Setting ECT(0) and ECT(1) Codepoints

New IETF specifications MAY permit a sender to set the ECT(0) or ECT(1) codepoint on a protocol control packet (including TCP segments for which [RFC3168] does not allow or recommend setting these codepoints.)

[AUTHORS' NOTE: Future versions of this document may take the form of such explicit text replacements.]

2.5. Clarification to the usage of the ECT(1) Codepoint

[RFC3168] notes that a router may treat and mark/drop packets differently depending on whether they observe the ECT(0) or ECT(1) codepoint. This specification permits new IETF specifications to set or read the ECT(1) codepeoint. It clarifies that these specifications do not necessarily treat ECT(1) as equivalent to

ECT(0).

Network devices using IETF-defined DSCPs MUST NOT re-mark packets to the ECT(1) codepoint. Specifically, the methods described in earlier ECN implementations using this codepoint as a congestion mark (described in Section 11.2.1 of [RFC3168]) are NOT RECOMMENDED for deployment in the current Internet.

3. Acknowledgements

The 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.

4. IANA Considerations

XX RFC ED - PLEASE REMOVE THIS SECTION XXX

This memo includes no request to IANA.

5. Security Considerations

The described method is a sender-side only transport change, and does not change the protocol messages exchanged. The security considerations of [RFC3168] therefore still apply.

A congestion control backoff that is less in response to ECN than the response to a packet loss can lead to a change in the capacity achieved when flows share a network bottleneck. This can result in redistribution of capacity between sharing flows, potentially resulting in unfairness in the way that capacity is shared. This potential gain applies only to ECN-marked packets using the updated method (and not to detected packet loss). Similar unfairness can be exhibited by congestion control mechanisms that have been used in the Internet for many years (e.g., CUBIC [I-D.CUBIC]). Unfairness may also be a result of other factors, including the round trip time experienced by a flow.

Packet loss can be expected from an AQM algorithm experiencing persistent queuing, but could also imply the presence of faulty equipment or media in a path, or it may imply the presence of congestion [RFC7567]. The update does not change the congestion control response to packet loss, and will therefore not lead to congestion collapse.

[AUTHORS' NOTE: Security considerations of the more relaxed rules of using ECT(0) vs. ECT(1) and usage of these ECT codepoints on any TCP segments will be included in the next version of this document.]

6. Revision Information

XX RFC ED - PLEASE REMOVE THIS SECTION XXX

-01. Broadened the scope to also cover ECT(0) vs. ECT(1) usage and using ECT(0) or ECT(1) codepoints on any TCP segments.

-00. draft-khademi-tsvwg-ecn-response-00 and draft-khademi-tcpm-alternativebackoff-ecn-00 replace draft-khademi-alternativebackoff-ecn-03, following discussion in the TSVWG and TCPM working groups.

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>.

[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>.

[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.ABE] Khademi, N., Welzl, M., Armitage, G., and G. Fairhurst, "TCP Alternative Backoff with ECN (ABE)", Internet-draft, IETF work-in-progress draft-khademi-tcpm-alternativebackoff-ecn-00, May 2016.

[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-08, November 2015.

[I-D.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-01, January 2016.

[I-D.CoDel] Nichols, K., Jacobson, V., McGregor, V., and J. Iyengar, "Controlled Delay Active Queue Management", Internet-draft, IETF work-in-progress draft-ietf-aqm-codel-03, March 2016.

[I-D.PIE] Pan, R., Natarajan, P., Baker, F., White, G., VerSteeg, B., Prabhu, M., Piglione, C., and V. Subramanian, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem", Internet-draft, IETF work-in-progress draft-ietf-aqm-pie-07, April 2016.

[I-D.bagnulo-tsvwg-generalized-ecn] Bagnulo, M. and B. Briscoe, "Adding Explicit Congestion Notification (ECN) to TCP control packets", draft-bagnulo-tsvwg-generalized-ecn-01 (work in progress), July 2016.

[I-D.briscoe-tsvwg-ecn-l4s-id]
Schepper, K., Briscoe, B., and I. Tsang, "Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay", draft-briscoe-tsvwg-ecn-l4s-id-01 (work in progress), March 2016.

[I-D.ietf-tcpm-accurate-ecn]
Briscoe, B., Kuehlewind, M., and R. Scheffenegger, "More Accurate ECN Feedback in TCP", draft-ietf-tcpm-accurate-ecn-00 (work in progress), December 2015.

[I-D.ietf-tcpm-dctcp]
Bensley, S., Eggert, L., Thaler, D., Balasubramanian, P., and G. Judd, "Datacenter TCP (DCTCP): TCP Congestion Control for Datacenters", draft-ietf-tcpm-dctcp-01 (work in progress), November 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>.

[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>.

[RFC7713] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements", RFC 7713, DOI 10.17487/RFC7713, December 2015, <http://www.rfc-editor.org/info/rfc7713>.

[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

TSVWG
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2017

V. Roca
INRIA
A. Begen
Networked Media

June 27, 2017

Forward Error Correction (FEC) Framework Extension to Sliding Window Codes
draft-roca-tsvwg-fecframev2-04

Abstract

RFC 6363 describes a framework for using Forward Error Correction (FEC) codes with applications in public and private IP networks to provide protection against packet loss. The framework supports applying FEC to arbitrary packet flows over unreliable transport and is primarily intended for real-time, or streaming, media. However FECFRAME as per RFC 6363 is restricted to block FEC codes. The present document extends FECFRAME to support FEC Codes based on a sliding encoding window, in addition to Block FEC Codes, in a backward compatible way. During multicast/broadcast real-time content delivery, the use of sliding window codes significantly improves robustness in harsh environments, with less repair traffic and lower FEC-related added latency.

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 December 29, 2017.

Copyright Notice

Table of Contents

1. Introduction

Many applications need to transport a continuous stream of packetized data from a source (sender) to one or more destinations (receivers) over networks that do not provide guaranteed packet delivery. In particular packets may be lost, which is strictly the focus of this document: we assume that transmitted packets are either received without any corruption or totally lost (e.g., because of a congested router, of a poor signal-to-noise ratio in a wireless network, or because the number of bit errors exceeds the correction capabilities of a low-layer error correcting code).

For these use-cases, Forward Error Correction (FEC) applied within the transport or application layer, is an efficient technique to improve packet transmission robustness in presence of packet losses (or "erasures"), without going through packet retransmissions that create a delay often incompatible with real-time constraints. The FEC Building Block defined in [RFC5052] provides a framework for the definition of Content Delivery Protocols (CDPs) that make use of separately defined FEC schemes. Any CDP defined according to the requirements of the FEC Building Block can then easily be used with any FEC scheme that is also defined according to the requirements of the FEC Building Block.

Then FECFRAME [RFC6363] provides a framework to define Content Delivery Protocols (CDPs) that provide FEC protection for arbitrary packet flows over unreliable transports such as UDP. It is primarily intended for real-time or streaming media applications, using broadcast, multicast, or on-demand delivery.

However [RFC6363] only considers block FEC schemes defined in accordance with the FEC Building Block [RFC5052] (e.g., [RFC6681], [RFC6816] or [RFC6865]). These codes require the input flow(s) to be segmented into a sequence of blocks. Then FEC encoding (at a sender or an encoding middlebox) and decoding (at a receiver or a decoding middlebox) are both performed on a per-block basis. This approach has major impacts on FEC encoding and decoding delays. The data packets of continuous media flow(s) can be sent immediately, without delay. But the block creation time, that depends on the number k of source symbols in this block, impacts the FEC encoding delay since encoding requires that all source symbols be known. This block creation time also impacts the decoding delay a receiver will experience in case of erasures, since no repair symbol for the current block can be received before. Therefore a good value for the block size is necessarily a balance between the maximum decoding latency at the receivers (which must be in line with the most stringent real-time requirement of the protected flow(s), hence an incentive to reduce the block size), and the desired robustness against long loss bursts (which increases with the block size, hence an incentive to increase this size).

This document extends [RFC6363] in order to also support FEC codes based on a sliding encoding window (A.K.A. convolutional codes). This encoding window, either of fixed or variable size, slides over the set of source symbols. FEC encoding is launched whenever needed, from the set of source symbols present in the sliding encoding window at that time. This approach significantly reduces FEC-related latency, since repair symbols can be generated and sent on-the-fly, at any time, and can be regularly received by receivers to quickly recover packet losses. Using sliding window FEC codes is therefore

highly beneficial to real-time flows, one of the primary targets of FECFRAME. [RLC-ID] provides an example of such FEC Scheme for FECFRAME, built upon the simple sliding window Random Linear Codes (RLC).

This document is fully backward compatible with [RFC6363] that it extends but does not replace. Indeed:

- this extension does not prevent nor compromize in any way the support of block FEC codes. Both types of codes can nicely co-exist, just like different block FEC schemes can co-exist;
- any receiver, for instance a legacy receiver that only supports block FEC schemes, can easily identify the FEC scheme used in a FECFRAME session thanks to the associated SDP file and its FEC Encoding ID information (i.e., the "encoding-id=" parameter of a "fec-repair-flow" attribute, [RFC6364]). This mechanism is not specific to this extension but is the basic approach for a FECFRAME receiver to determine whether or not it supports the FEC scheme used in a given FECFRAME session;

This document leverages on [RFC6363] and re-uses its structure. It proposes new sections specific to sliding window FEC codes whenever required. The only exception is Section Section 3 that provides a quick summary of FECFRAME in order to facilitate the understanding of this document to readers not familiar with the concepts and terminology.

2. Definitions and Abbreviations

The following list of definitions and abbreviations is copied from [RFC6363], adding only the Block/sliding window FEC Code and Encoding/Decoding Window definitions:

Application Data Unit (ADU): The unit of source data provided as payload to the transport layer.

ADU Flow: A sequence of ADUs associated with a transport-layer flow identifier (such as the standard 5-tuple {source IP address, source port, destination IP address, destination port, transport protocol}).

AL-FEC: Application-layer Forward Error Correction.

Application Protocol: Control protocol used to establish and control the source flow being protected, e.g., the Real-Time Streaming Protocol (RTSP).

Content Delivery Protocol (CDP): A complete application protocol specification that, through the use of the framework defined in this document, is able to make use of FEC schemes to provide FEC capabilities.

FEC Code: An algorithm for encoding data such that the encoded data flow is resilient to data loss. Note that, in general, FEC codes may also be used to make a data flow resilient to corruption, but that is not considered in this document.

Block FEC Code: An FEC Code that operates in a block manner, i.e., for which the input flow MUST be segmented into a sequence of blocks, FEC encoding and decoding being performed independently on a per-block basis.

Sliding Window (or Convolutional) FEC Code: An FEC Code that can generate repair symbols on-the-fly, at any time, from the set of source symbols present in the sliding encoding window at that time.

FEC Framework: A protocol framework for the definition of Content Delivery Protocols using FEC, such as the framework defined in this document.

FEC Framework Configuration Information: Information that controls the operation of the FEC Framework.

FEC Payload ID: Information that identifies the contents of a packet with respect to the FEC scheme.

FEC Repair Packet: At a sender (respectively, at a receiver), a payload submitted to (respectively, received from) the transport protocol containing one or more repair symbols along with a Repair FEC Payload ID and possibly an RTP header.

FEC Scheme: A specification that defines the additional protocol aspects required to use a particular FEC code with the FEC Framework.

FEC Source Packet: At a sender (respectively, at a receiver), a payload submitted to (respectively, received from) the transport protocol containing an ADU along with an optional Explicit Source FEC Payload ID.

Protection Amount: The relative increase in data sent due to the use of FEC.

Repair Flow: The packet flow carrying FEC data.

Repair FEC Payload ID: A FEC Payload ID specifically for use with repair packets.

Source Flow: The packet flow to which FEC protection is to be applied. A source flow consists of ADUs.

Source FEC Payload ID: A FEC Payload ID specifically for use with source packets.

Source Protocol: A protocol used for the source flow being protected, e.g., RTP.

Transport Protocol: The protocol used for the transport of the source and repair flows, e.g., UDP and the Datagram Congestion Control Protocol (DCCP).

Encoding Window: Set of Source Symbols available at the sender/ coding node that are used to generate a repair symbol, with a Sliding Window FEC Code.

Decoding Window: Set of received or decoded source and repair symbols available at a receiver that are used to decode erased source symbols, with a Sliding Window FEC Code.

Code Rate: The ratio between the number of source symbols and the number of encoding symbols. By definition, the code rate is such that 0 < code rate <= 1. A code rate close to 1 indicates that a small number of repair symbols have been produced during the encoding process.

Encoding Symbol: Unit of data generated by the encoding process. With systematic codes, source symbols are part of the encoding symbols.

Packet Erasure Channel: A communication path where packets are either lost (e.g., by a congested router, or because the number of transmission errors exceeds the correction capabilities of the physical-layer codes) or received. When a packet is received, it is assumed that this packet is not corrupted.

Repair Symbol: Encoding symbol that is not a source symbol.

Source Block: Group of ADUs that are to be FEC protected as a single block. This notion is restricted to Block FEC Codes.

Source Symbol: Unit of data used during the encoding process.

Systematic Code: FEC code in which the source symbols are part of the encoding symbols.

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. Architecture Overview

The architecture of [RFC6363], Section 3, equally applies to this FECFRAME extension and is not repeated here. However we provide hereafter a quick summary to facilitate the understanding of this document to readers not familiar with the concepts and terminology.

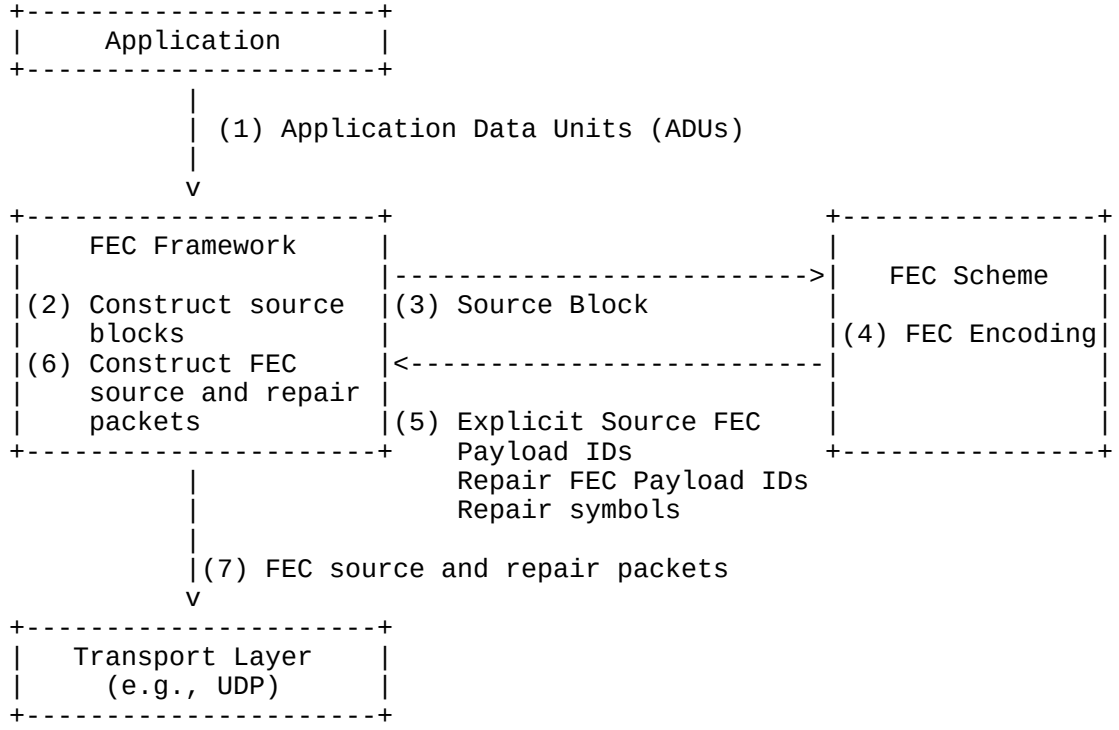


Figure 1: FECFRAME architecture at a sender.

The FECFRAME architecture is illustrated in Figure 1 from the sender's point of view, in case of a block FEC Scheme. It shows an application generating an ADU flow (other flows, from other applications, may co-exist). These ADUs, of variable size, must be somehow mapped to source symbols of fixed size. This is the goal of an ADU to symbols mapping process that is FEC Scheme specific (see

below). Once the source block is built, taking into account both the FEC Scheme constraints (e.g., in terms of maximum source block size) and the application's flow constraints (e.g., real-time constraints), the associated source symbols are handed to the FEC Scheme in order to produce an appropriate number of repair symbols. FEC Source Packets (containing ADUs) and FEC Repair Packets (containing one or more repair symbols each) are then generated and sent using UDP (more precisely [RFC6363], Section 7, requires a transport protocol providing an unreliable datagram service, like UDP or DCCP). In practice FEC Source Packets can be sent as soon as available, without having to wait for FEC encoding to take place. In that case a copy of the associated source symbols need to be kept within FECFRAME for future FEC encoding purposes.

At a receiver (not shown), FECFRAME processing operates in a similar way, taking as input the incoming FEC source and repair packets received. In case of FEC source packet losses, when the FEC decoding of the associated block recovers all the missing source symbols, the lost ADUs are recovered and assigned to their respective flow (see below). ADUs are then returned to the application(s), either in order or not depending on the application requirements.

FECFRAME features two subtle mechanisms:

- ADUs to source symbols mapping: in order to manage variable size ADUs, FECFRAME and FEC Schemes can use small, fixed size, symbols and create a mapping between ADUs and symbols. To each ADU this mechanism prepends a length field (plus a flow identifier, see below) and pads the result to a multiple of the symbol size. A small ADU may be mapped to a single source symbol while a large one may be mapped to multiple symbols. The mapping details are FEC Scheme dependant and must be defined there.

- Assignment of decoded ADUs to flows in multi-flow configurations: when multiple flows are multiplexed over the same FECFRAME instance, a problem is to assign a decoded ADU to the right flow (UDP port numbers/IP addresses traditionally used to map incoming ADUs to flows are not recovered during FEC decoding). To make it possible, at the FECFRAME sending instance, each ADU is prepended with a flow identifier (1 byte) before doing the mapping to source symbols (see above). This (flow ID + length + application payload + padding), called ADUI, is then FEC protected. Therefore a decoded ADUI contains enough information to assign the ADU to the right flow.

A few aspects are not considered by FECFRAME, namely:

- o congestion control (see [RFC6363], section 8 for a more detailed discussion);
- o feedbacks from receiver(s) (although they may exist within the application, e.g., through RCTP control messages);
- o flow adaptation at a FECFRAME sender (e.g., by adjusting the FEC code rate based on channel conditions, since there is no feedback mechanism within FECFRAME);

4. Procedural Overview

4.1. General

The general considerations of [RFC6363], Section 4.1, that are specific to block FEC codes are not repeated here.

With a Sliding Window FEC Code, the FEC source packet MUST contain information to identify the position occupied by the ADU within the source flow, in terms specific to the FEC scheme. This information is known as the Source FEC Payload ID, and the FEC scheme is responsible for defining and interpreting it.

With a Sliding Window FEC Code, the FEC repair packets MUST contain information that identifies the relationship between the contained repair payloads and the original source symbols used during encoding. This information is known as the Repair FEC Payload ID, and the FEC scheme is responsible for defining and interpreting it.

The Sender Operation ([RFC6363], Section 4.2.) and Receiver Operation ([RFC6363], Section 4.3) are both specific to block FEC codes and therefore omitted below. The following two sections detail similar operations for Sliding Window FEC codes.

4.2. Sender Operation with Sliding Window FEC Codes

With a Sliding Window FEC scheme, the following operations, illustrated in Figure 2 for the case of UDP repair flows, and in Figure 3 for the case of RTP repair flows, describe a possible way to generate compliant source and repair flows:

1. A new ADU is provided by the application.
2. The FEC Framework communicates this ADU to the FEC scheme.
3. The sliding encoding window is updated by the FEC scheme. The ADU to source symbols mapping as well as the encoding window management details are both the responsibility of the FEC scheme

and MUST be detailed there. Appendix A provides some hints on the way it might be performed.

4. The Source FEC Payload ID information of the source packet is determined by the FEC scheme. If required by the FEC scheme, the Source FEC Payload ID is encoded into the Explicit Source FEC Payload ID field and returned to the FEC Framework.

5. The FEC Framework constructs the FEC source packet according to [RFC6363] Figure 6, using the Explicit Source FEC Payload ID provided by the FEC scheme if applicable.

6. The FEC source packet is sent using normal transport-layer procedures. This packet is sent using the same ADU flow identification information as would have been used for the original source packet if the FEC Framework were not present (for example, in the UDP case, the UDP source and destination addresses and ports on the IP datagram carrying the source packet will be the same whether or not the FEC Framework is applied).

7. When the FEC Framework needs to send one or several FEC repair packets (e.g., according to the target Code Rate), it asks the FEC scheme to create one or several repair packet payloads from the current sliding encoding window along with their Repair FEC Payload ID.

8. The Repair FEC Payload IDs and repair packet payloads are provided back by the FEC scheme to the FEC Framework.

9. The FEC Framework constructs FEC repair packets according to [RFC6363] Figure 7, using the FEC Payload IDs and repair packet payloads provided by the FEC scheme.

10. The FEC repair packets are sent using normal transport-layer procedures. The port(s) and multicast group(s) to be used for FEC repair packets are defined in the FEC Framework Configuration Information.

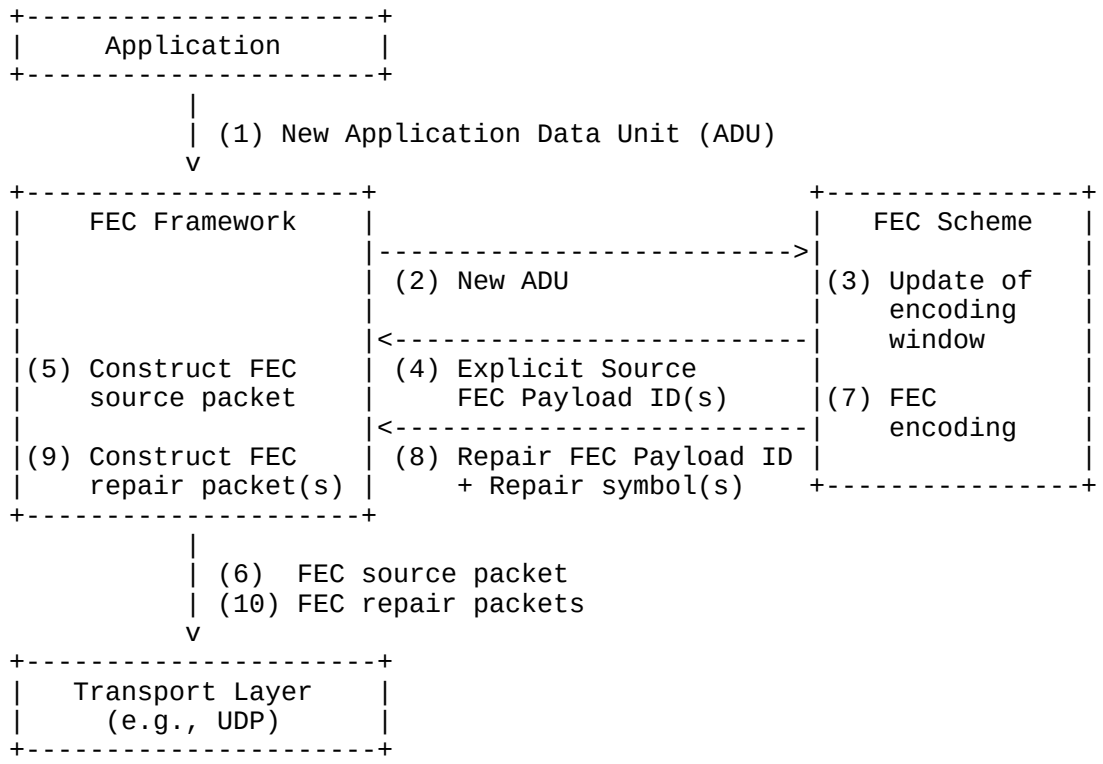


Figure 2: Sender Operation with Convolutional FEC Codes

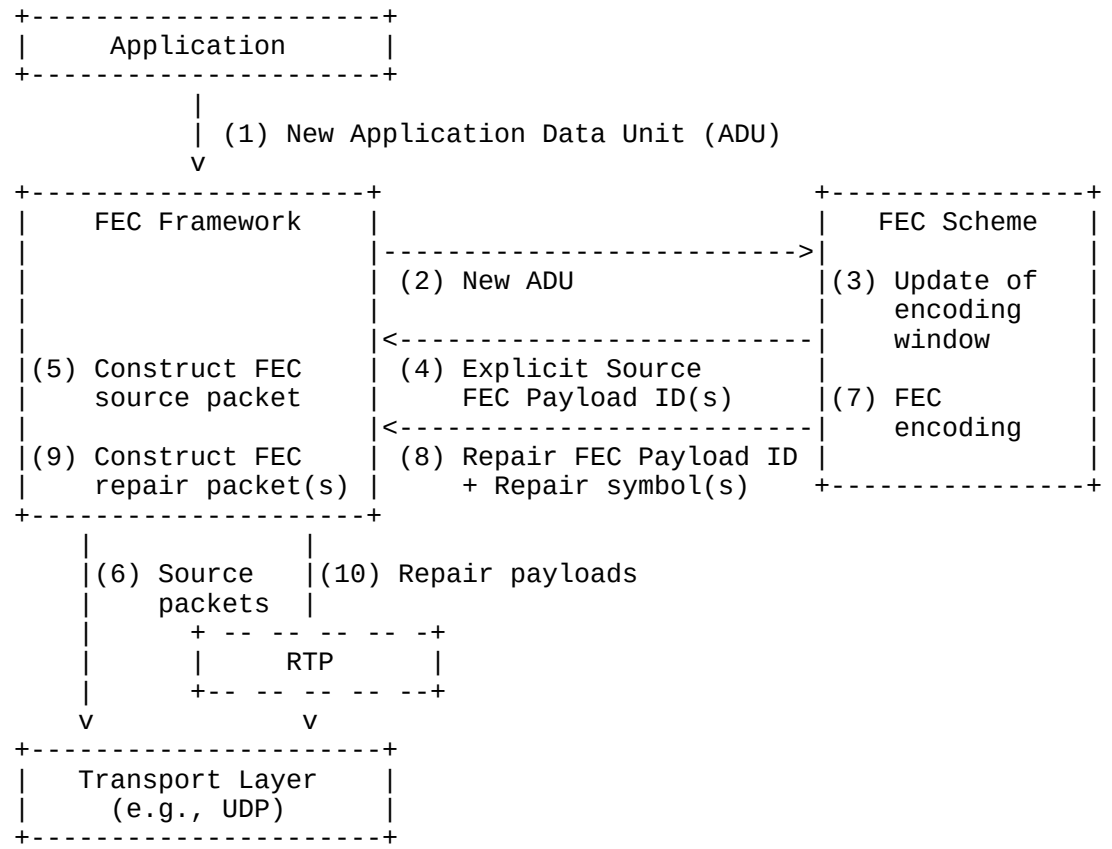


Figure 3: Sender Operation with RTP Repair Flows

4.3. Receiver Operation with Sliding Window FEC Codes

With a Sliding Window FEC scheme, the following operations, illustrated in Figure 4 for the case of UDP repair flows, and in Figure 5 for the case of RTP repair flows. The only differences with respect to block FEC codes lie in steps (4) and (5). Therefore this section does not repeat the other steps of [RFC6363], Section 4.3, "Receiver Operation". The new steps (4) and (5) are:

4. The FEC scheme uses the received FEC Payload IDs (and derived FEC Source Payload IDs when the Explicit Source FEC Payload ID field is not used) to insert source and repair packets into the decoding window in the right way. If at least one source packet is missing and at least one repair packet has been received and the rank of the associated linear system permits it, then FEC decoding can be performed in order to recover missing source

payloads. The FEC scheme determines whether source packets have been lost and whether enough repair packets have been received to decode any or all of the missing source payloads.

5. The FEC scheme returns the received and decoded ADUs to the FEC Framework, along with indications of any ADUs that were missing and could not be decoded.

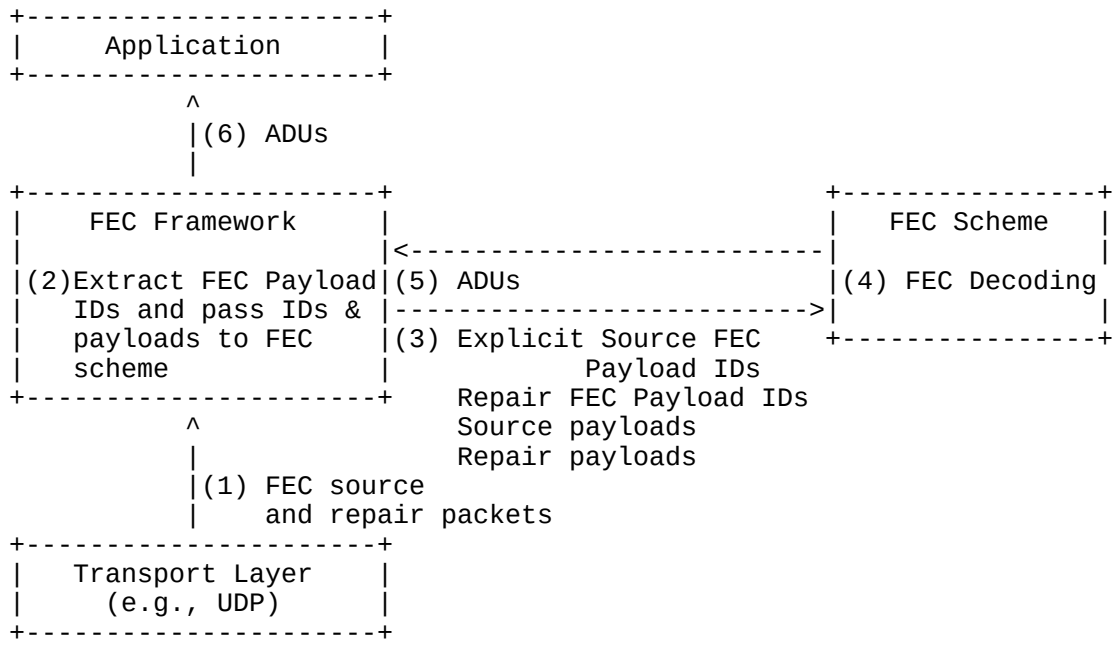


Figure 4: Receiver Operation with Sliding Window FEC Codes

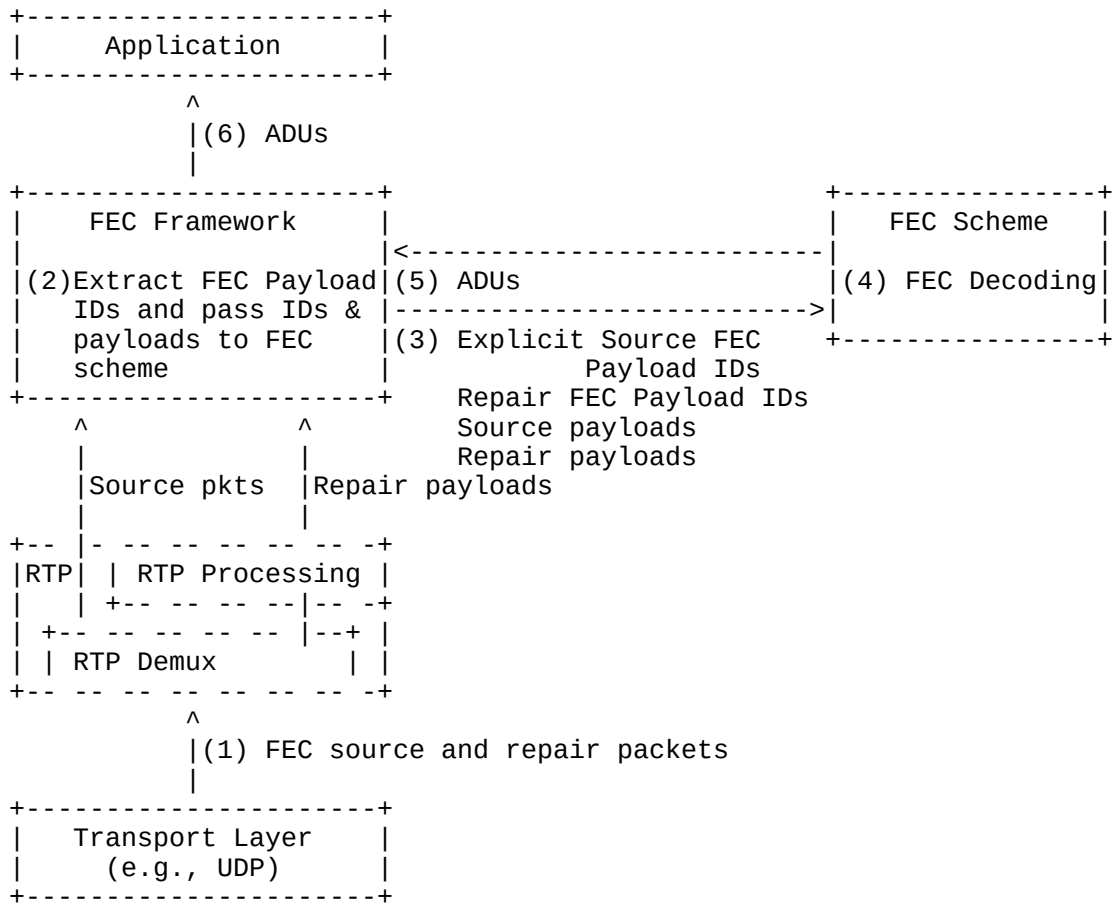


Figure 5: Receiver Operation with RTP Repair Flows

5. Protocol Specification

5.1. General

This section discusses the protocol elements for the FEC Framework specific to Sliding Window FEC schemes. The global formats of source data packets (i.e., [RFC6363], Figure 6) and repair data packets (i.e., [RFC6363], Figures 7 and 8) remain the same with Sliding Window FEC codes. They are not repeated here.

5.2. FEC Framework Configuration Information

The FEC Framework Configuration Information considerations of [RFC6363], Section 5.5, equally applies to this FECFRAME extension and is not repeated here.

5.3. FEC Scheme Requirements

The FEC scheme requirements of [RFC6363], Section 5.6, mostly apply to this FECFRAME extension and are not repeated here. An exception though is the "full specification of the FEC code", item (4), that is specific to block FEC codes. The following item (4) applies instead:

4. A full specification of the Sliding Window FEC code

   This specification MUST precisely define the valid FEC-Scheme-Specific Information values, the valid FEC Payload ID values, and the valid packet payload sizes (where packet payload refers to the space within a packet dedicated to carrying encoding symbols).

   Furthermore, given valid values of the FEC-Scheme-Specific Information, a valid Repair FEC Payload ID value, a valid packet payload size, and a valid encoding window (i.e., a set of source symbols), the specification MUST uniquely define the values of the encoding symbols to be included in the repair packet payload with the given Repair FEC Payload ID value.

Additionally, the FEC scheme associated to a Sliding Window FEC Code:

- MUST define the relationships between ADUs and the associated source symbols (mapping);
- MUST define the management of the encoding window that slides over the set of ADUs. Appendix A provides a non normative example;
- MUST define the management of the decoding window, consisting of a system of linear equations (in case of a linear FEC code);

6. Feedback

The discussion of [RFC6363], Section 6, equally applies to this FECFRAME extension and is not repeated here.

7. Transport Protocols

The discussion of [RFC6363], Section 7, equally applies to this FECFRAME extension and is not repeated here.

8. Congestion Control

The discussion of [RFC6363], Section 8, equally applies to this FECFRAME extension and is not repeated here.

9. Implementation Status

Editor's notes: RFC Editor, please remove this section motivated by RFC 7942 before publishing the RFC. Thanks!

An implementation of FECFRAME extended to Sliding Window codes exists:

- Organisation: Inria
- Description: This is an implementation of FECFRAME extended to Sliding Window codes and supporting the RLC FEC Scheme [RLC-ID]. It is based on: (1) a proprietary implementation of FECFRAME, made by Inria and Expway for which interoperability tests have been conducted; and (2) a proprietary implementation of RLC Sliding Window FEC Codes.
- Maturity: the basic FECFRAME maturity is "production", the FECFRAME extension maturity is "under progress".
- Coverage: the software implements a subset of [RFC6363], as specialized by the 3GPP eMBMS standard [MBMSTS]. This software also covers the additional features of FECFRAME extended to Sliding Window codes, in particular the RLC FEC Scheme.
- Lincensing: proprietary.
- Implementation experience: maximum.
- Information update date: March 2017.
- Contact: vincent.roca@inria.fr

10. Security Considerations

This FECFRAME extension does not add any new security consideration. All the considerations of [RFC6363], Section 9, apply to this document as well.

11. Operations and Management Considerations

This FECFRAME extension does not add any new Operations and Management Consideration. All the considerations of [RFC6363], Section 10, apply to this document as well.

12. IANA Considerations

A FEC scheme for use with this FEC Framework is identified via its FEC Encoding ID. It is subject to IANA registration in the "FEC Framework (FECFRAME) FEC Encoding IDs" registry. All the rules of [RFC6363], Section 11, apply and are not repeated here.

13. Acknowledgments

TBD

14. References

14.1. 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>.

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, DOI 10.17487/RFC6363, October 2011, <http://www.rfc-editor.org/info/rfc6363>.

14.2. Informative References

[MBMSTS] 3GPP, "Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs", 3GPP TS 26.346, March 2009, <http://ftp.3gpp.org/specs/html-info/26346.htm>.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, DOI 10.17487/RFC5052, August 2007, <http://www.rfc-editor.org/info/rfc5052>.

[RFC6364] Begen, A., "Session Description Protocol Elements for the Forward Error Correction (FEC) Framework", RFC 6364, DOI 10.17487/RFC6364, October 2011, <http://www.rfc-editor.org/info/rfc6364>.

[RFC6681] Watson, M., Stockhammer, T., and M. Luby, "Raptor Forward Error Correction (FEC) Schemes for FECFRAME", RFC 6681, DOI 10.17487/RFC6681, August 2012, <http://www.rfc-editor.org/info/rfc6681>.

[RFC6816] Roca, V., Cunche, M., and J. Lacan, "Simple Low-Density Parity Check (LDPC) Staircase Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6816, DOI 10.17487/RFC6816, December 2012, <http://www.rfc-editor.org/info/rfc6816>.

[RFC6865] Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., and K. Matsuzono, "Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6865, DOI 10.17487/RFC6865, February 2013, <http://www.rfc-editor.org/info/rfc6865>.

[RLC-ID] Roca, V., "The Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Scheme for FECFRAME", Work in Progress, Transport Area Working Group (TSVWG) draft-roca-tsvwg-rlc-fec-scheme (Work in Progress), June 2017, <https://tools.ietf.org/html/draft-roca-tsvwg-rlc-fec-scheme>.

Appendix A. About Sliding Encoding Window Management (non Normative)

The FEC Framework does not specify the management of the sliding encoding window which is the responsibility of the FEC Scheme. This annex provides a few hints with respect to the management of this encoding window.

Source symbols are added to the sliding encoding window each time a new ADU arrives, where the following information is provided for this ADU by the FEC Framework: a description of the source flow with which the ADU is associated, the ADU itself, and the length of the ADU. This information is sufficient for the FEC scheme to map the ADU to the corresponding source symbols.

Source symbols and the corresponding ADUs are removed from the sliding encoding window, for instance:

- after a certain delay, when an "old" ADU of a real-time flow times out. The source symbol retention delay in the sliding encoding window should therefore be initialized according to the real-time features of incoming flow(s).

- once the sliding encoding window has reached its maximum size (there is usually an upper limit to the sliding encoding window size). In that case the oldest symbol is removed each time a new source symbol is added.

Several aspects exist that can impact the sliding encoding window management:

- at the source flows level: real-time constraints can limit the total time source symbols can remain in the encoding window;

- at the FEC code level: there may be theoretical or practical limitations (e.g., because of computational complexity) that limit the number of source symbols in the encoding window.

- at the FEC scheme level: signaling and window management are intrinsically related. For instance, an encoding window composed of a non sequential set of source symbols requires an appropriate signaling to inform a receiver of the composition of the encoding window. On the opposite, an encoding window always composed of a sequential set of source symbols simplifies signaling: providing the identity of the first source symbol plus their number is sufficient.

Authors' Addresses

Vincent Roca
INRIA
Grenoble
France

EMail: vincent.roca@inria.fr

Ali Begen
Networked Media
Konya
Turkey

EMail: ali.begen@networked.media

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

This Internet-Draft will expire on November 16, 2017.

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".

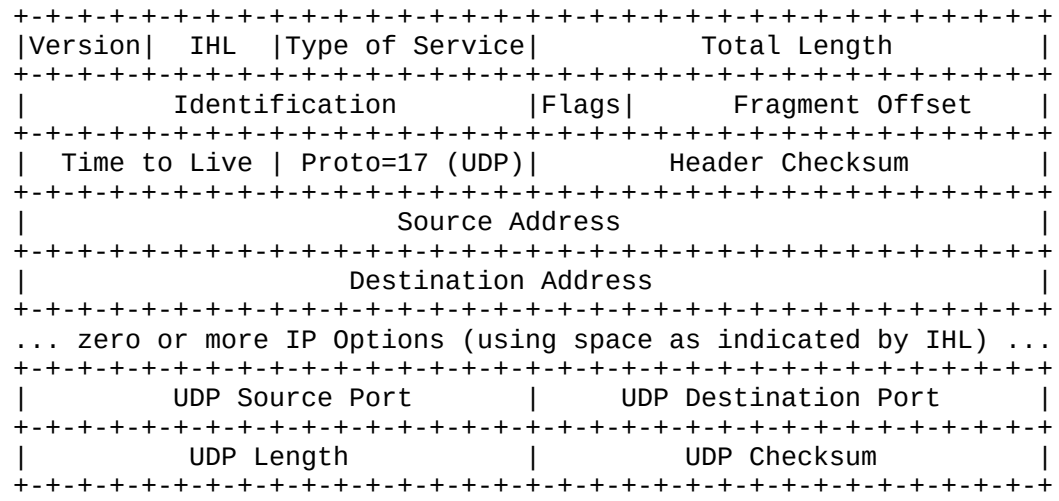


Figure 1 IPv4 datagram with UDP transport payload

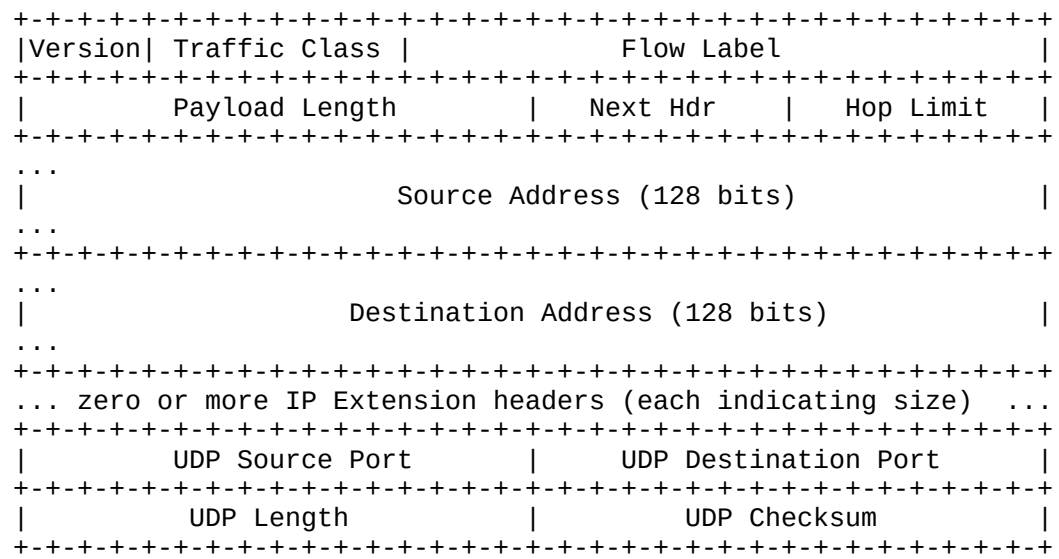


Figure 2 IPv6 datagram with UDP transport payload

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).

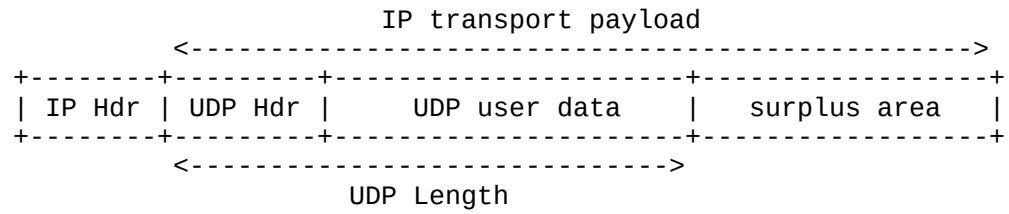


Figure 3 IP transport payload vs. UDP Length

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.

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.

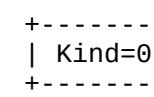


Figure 5 UDP EOL option format

>> 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.

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.

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.

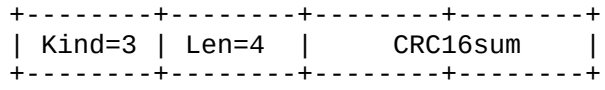


Figure 8 UDP ACS option format

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.

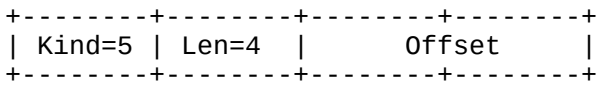


Figure 9 UDP LITE option format

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).

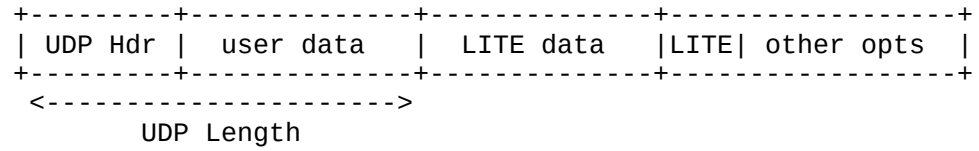


Figure 10 LITE option formation - LITE goes first

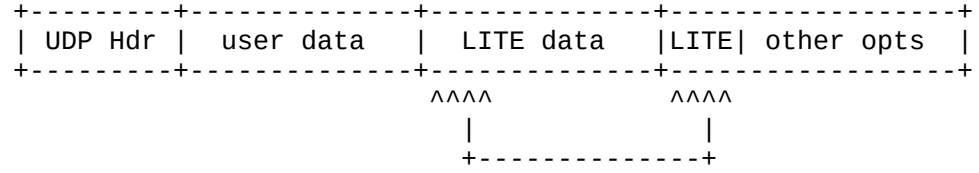


Figure 11 Before sending swap LITE option and front of LITE data

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.

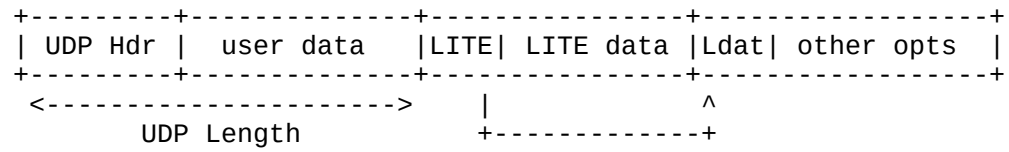


Figure 12 Lite option as sent

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].

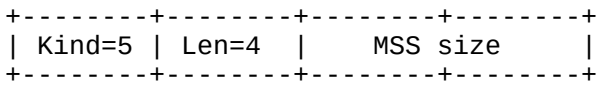


Figure 13 UDP MSS option format

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].

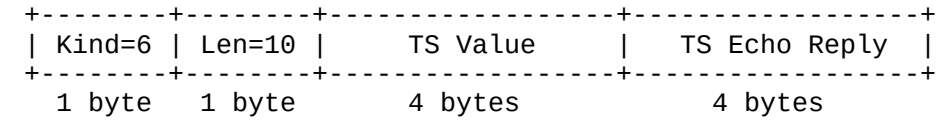


Figure 14 UDP TIME option format

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.

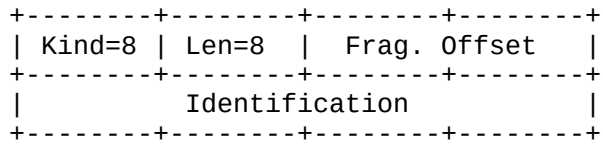


Figure 15 UDP non-terminal FRAG option format

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).

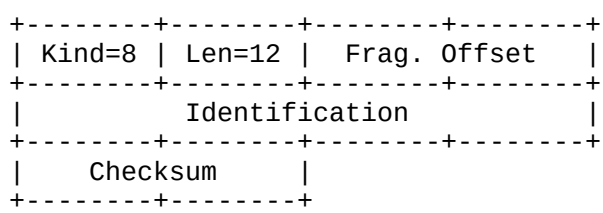


Figure 16 UDP terminal FRAG option format

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.

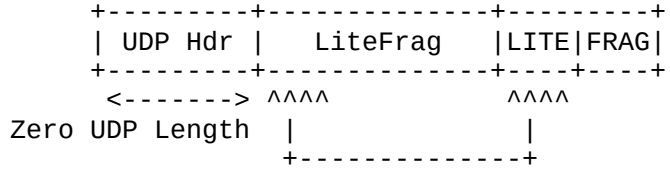


Figure 17 Preparing FRAG as Lite data

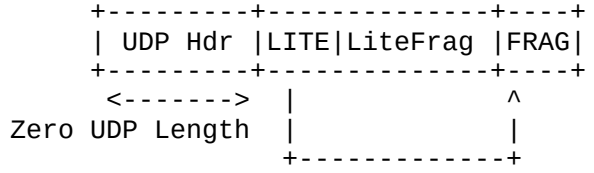


Figure 18 Lite option before transmission

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:

- This document defines the meaning of the IP payload area beyond the UDP length but within the IP length.

- 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: 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:

- The problem description,
- The text quoted from [RFC4960],
- The replacement text that should be placed into an upcoming BIS document,
- 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)

- 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)

- 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]:

- 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.
- 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)

- 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)

- 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:

- SCTP MAY increment cwnd by 1*MTU.
- 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.
- SCTP MUST NOT increment cwnd by more than 1*MTU per RTT.

Old text: (Section 7.2.2)

- 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.
- 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)

- 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.

- 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.

- 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)

- 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)

- The initial cwnd before DATA transmission MUST be set to min(4*MTU, max (2*MTU, 4380 bytes)).

Old text: (Section 7.2.1)

- 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)

- 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:

- 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.

- 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

Network Working Group
Internet-Draft
Updates: 6951 (if approved)
Intended status: Standards Track
Expires: 2 April 2021

M. Tüxen
Münster Univ. of Appl. Sciences
R. R. Stewart
Netflix, Inc.

29 September 2020

Additional Considerations for UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets
draft-tuexen-tsvwg-sctp-udp-encaps-cons-03

Abstract

RFC 6951 specifies the UDP encapsulation of SCTP packets. The described handling of received packets requires the check of the verification tag. However, RFC 6951 misses a specification for the handling of received packets for which this check is not possible.

This document updates RFC 6951 by specifying the handling of received packets where the verification tag can not be checked.

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 https://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 2 April 2021.

Copyright Notice

Table of Contents

1. Introduction

[RFC6951] specifies the UDP encapsulation of SCTP packets. To be able to adopt automatically to changes of the remote UDP encapsulation port number, it is updated automatically when processing received packets. This includes automatic enabling and disabling of UDP encapsulation.

Section 5.4 of [RFC6951] describes the processing of received packets and requires the check of the verification tag before updating the remote UDP encapsulation port and the possible enabling or disabling of UDP encapsulation.

[RFC6951] basically misses a description of the handling of received packets where checking the verification tag is not possible. This includes packets for which no association can be found and packets containing an INIT chunk, since the verification tag of these packets is 0.

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. Handling of Out of the Blue Packets

If the processing of an out of the blue packet requires the sending of a packet in response according to the rules specified in Section 8.4 of [RFC4960], the following rules apply:

1. If the received packet was encapsulated in UDP, the response packets MUST also be encapsulated in UDP. The UDP source port and UDP destination port used for sending the response packet are the UDP destination port and UDP source port of the received packet.

2. If the receive packet was not encapsulated in UDP, the response packet MUST NOT be encapsulated in UDP.

Please not that in these cases a check of the of the verification tag is not possible.

4. Handling of SCTP Packets Containing an INIT Chunk Matching an Existing Associations

SCTP packets containing an INIT chunk have the verification tag 0 in the common header. Therefore the verification can’t be checked.

The following rules apply when processing the received packet:

1. The remote UDP encapsulation port for the source address of the received SCTP packet MUST NOT be updated if the encapsulation of outgoing packets is enabled and the received SCTP packet is encapsulated.

2. The UDP encapsulation for outgoing packets towards the source address of the received SCTP packet MUST NOT be enabled, if it is disabled and the received SCTP packet is encapsulated.

3. The UDP encapsulation for outgoing packets towards the source address of the received SCTP packet MUST NOT be disabled, if it is enabled and the received SCTP packet is not encapsulated.

4. If the UDP encapsulation for outgoing packets towards the source address of the received SCTP packet is disabled and the received SCTP packet is encapsulated, an SCTP packet containing an ABORT chunk MUST be sent. The ABORT chunk MAY include the error cause defined below indicating an "Restart of an Association with New Encapsulation Port". This packet containing the ABORT chunk MUST be encapsulated in UDP. The UDP source port and UDP destination port used for sending the packet containing the ABORT chunk are the UDP destination port and UDP source port of the received packet containing the INIT chunk.

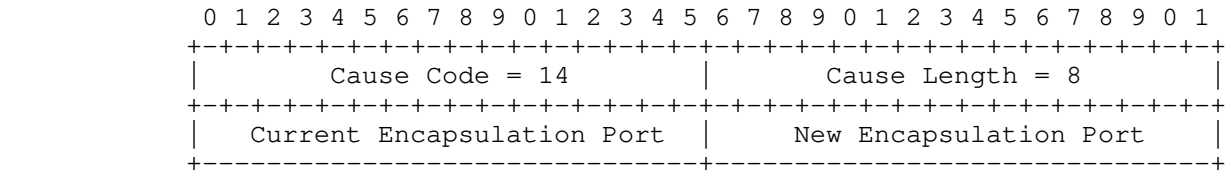
5. If the UDP encapsulation for outgoing packets towards the source address of the received SCTP packet is disabled and the received SCTP packet is not encapsulated, the processing defined in [RFC4960] MUST be performed. If a packet is sent in response, it MUST NOT be encapsulated.

6. If the UDP encapsulation for outgoing packets towards the source address of the received SCTP packet is enabled and the received SCTP packet is not encapsulated, an SCTP packet containing an ABORT chunk MUST be sent. The ABORT chunk MAY include the error cause defined below indicating an "Restart of an Association with New Encapsulation Port". This packet containing the ABORT chunk MUST NOT be encapsulated in UDP.

7. If the UDP encapsulation for outgoing packets towards the source address of the received SCTP packet is enabled and the received SCTP packet is encapsulated, but the UDP source port of the received SCTP packet is not equal to the remote UDP encapsulation port for the source address of the received SCTP packet, an SCTP packet containing an ABORT chunk MUST be sent. The ABORT chunk MAY include the error cause defined below indicating an "Restart of an Association with New Encapsulation Port". This packet containing the ABORT chunk MUST be encapsulated in UDP. The UDP source port and UDP destination port used for sending the packet containing the ABORT chunk are the UDP destination port and UDP source port of the received packet containing the INIT chunk.

8. If the UDP encapsulation for outgoing packets towards the source address of the received SCTP packet is enabled and the received SCTP packet is encapsulated and the UDP source port of the received SCTP packet is equal to the remote UDP encapsulation port for the source address of the received SCTP packet, the processing defined in [RFC4960] MUST be performed. If a packet is sent in response, it MUST be encapsulated. The UDP source port and UDP destination port used for sending the packet containing the ABORT chunk are the UDP destination port and UDP source port of the received packet containing the INIT chunk.

The error cause indicating an "Restart of an Association with New Encapsulation Port" is defined by the following figure.



Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the "Restart of an Association with New Encapsulation Port" error cause. IANA is requested to assign the value 14 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 8.

Current Encapsulation Port: 2 bytes (unsigned integer)
This field holds the remote encapsulation port currently being used for the destination address the received packet containing the INIT chunk was sent from. If the UDP encapsulation for destination address is currently disabled, 0 is used.

New Encapsulation Port: 2 bytes (unsigned integer)
If the received SCTP packet containing the INIT chunk is encapsulated in UDP, this field holds the UDP source port number of the UDP packet. If the received SCTP packet is not encapsulated in UDP, this field is 0.

All transported integer numbers are in "network byte order" a.k.a., Big Endian.

5. Middlebox Considerations

Middleboxes often use different timeouts for UDP based flows than for other flows. Therefore the HEARTBEAT.Interval parameter SHOULD be lowered to 15 seconds when UDP encapsulation is used.

6. 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 the registration described in this section.

A new error cause code has to be assigned by IANA. This requires an additional line in the "Error Cause Codes" registry for SCTP:

Error Cause Codes

Value	Cause Code	Reference
14	Restart of an Association with New Encapsulation Port	[RFCXXXX]

Table 1

7. Security Considerations

This document does not change the considerations given in [RFC6951].

However, not following the procedures given in this document might allow an attacker to take over SCTP associations. The attacker needs only to share the IP address of an existing SCTP association.

It should also be noted that if firewalls will be applied at the SCTP association level they have to take the UDP encapsulation into account.

8. Acknowledgments

The authors wish to thank Georgios Papastergiou for the initial problem report.

The authors wish to thank Irene Rüngeler and Felix Weinrank 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).

9. 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>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[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>.

[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>.

Authors' Addresses

Michael Tüxen
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
United States

Email: randall@lakerest.net