

Transport Area Working Group
Internet-Draft
Obsoletes: 3540 (if approved)
Updates: 3168, 4341, 4342, 5622, 6679
(if approved)
Intended status: Standards Track
Expires: May 21, 2017

D. Black
Dell EMC
November 17, 2016

Explicit Congestion Notification (ECN) Experimentation
draft-black-tsvwg-ecn-experimentation-04

Abstract

Multiple protocol experiments have been proposed that involve changes to Explicit Congestion Notification (ECN) as specified in RFC 3168. This memo summarizes the proposed areas of experimentation to provide an overview to the Internet community and updates RFC 3168, a Proposed Standard RFC, to allow the experiments to proceed without requiring a standards process exception for each Experimental RFC to update RFC 3168. Each experiment is still required to be documented in an Experimental RFC. In addition, this memo makes related updates to the ECN specifications for RTP in RFC 6679 and to the ECN specifications for DCCP in RFC 4341, RFC 4342 and RFC 5622. This memo also records the conclusion of the ECN Nonce experiment in RFC 3540, obsoletes RFC 3540 and reclassifies it as Historic to enable new experimental use of the ECT(1) codepoint.

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 May 21, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. Scope of ECN Experiments	3
3. ECN Nonce and RFC 3540	4
4. Updates to RFC 3168	5
4.1. Congestion Response Differences	5
4.2. ECT Differences	6
4.3. Generalized ECN	7
4.4. Effective Congestion Control is Required	8
5. ECN for RTP Updates to RFC 6679	9
6. ECN for DCCP Updates to RFCs 4341, 4342 and 5622	10
7. Acknowledgements	10
8. IANA Considerations	11
9. Security Considerations	11
10. References	11
10.1. Normative References	11
10.2. Informative References	12
Author's Address	14

1. Introduction

Multiple protocol experiments have been proposed that involve changes to Explicit Congestion Notification (ECN) as specified in RFC 3168 [RFC3168]. This memo summarizes the proposed areas of experimentation to provide an overview to the Internet community and updates RFC 3168 to allow the experiments to proceed without requiring a standards process exception for each Experimental RFC to update RFC 3168, a Proposed Standard RFC. This memo also makes related updates to the ECN specification for RTP in RFC 6679 [RFC6679] for the same reason. Each experiment is still required to be documented in one or more separate RFCs, but use of Experimental RFCs for this purpose does not require a process exception to modify RFC 3168 or RFC 6679 when the modification falls within the bounds established by this memo.

One of these areas of experimentation involves use of the ECT(1) codepoint that was dedicated to the ECN Nonce experiment as described in RFC 3540 [RFC3540]. This memo records the conclusion of the ECN Nonce experiment, obsoletes RFC 3540 and reclassifies it as Historic.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

2. Scope of ECN Experiments

Three areas of ECN experimentation are covered by this memo; the cited Internet-Drafts should be consulted for the goals and rationale of each proposed experiment:

Congestion Response Differences: For congestion indicated by ECN, use a different IETF-approved sender congestion response (e.g., reduce the response so that the sender backs off by a smaller amount) by comparison to congestion indicated by loss, e.g., as proposed in [I-D.khademi-tcpm-alternativebackoff-ecn] and [I-D.briscoe-tsvwg-ecn-l4s-id] - the experiment in the latter draft couples the backoff change to ECT Differences functionality (next bullet). This is at variance with RFC 3168's requirement that a sender's congestion control response to ECN congestion indications be the same as to drops.

ECT Differences: Use ECT(1) to request ECN congestion marking behavior in the network that differs from ECT(0) counterbalanced by use of a different IETF-approved congestion response to CE

marks at the sender, e.g., as proposed in [I-D.briscoe-tsvwg-ecn-l4s-id]. This is at variance with RFC 3168's requirement that ECT(0)-marked traffic and ECT(1)-marked traffic not receive different treatment in the network.

Generalized ECN: Use ECN for TCP control packets (i.e., send control packets such as SYN with ECT marking) and for retransmitted packets, e.g., as proposed in [I-D.bagnulo-tsvwg-generalized-ecn]. This is at variance with RFC 3168's prohibition of use of ECN for TCP control packets and retransmitted packets

The scope of this memo is limited to these three areas of experimentation. This memo neither prejudices the outcomes of the proposed experiments nor specifies the experiments in detail. The purpose of this memo is to remove constraints in standards track RFCs that serve to prohibit these areas of experimentation.

3. ECN Nonce and RFC 3540

As specified in RFC 3168, ECN uses two ECN Capable Transport (ECT) codepoints to indicate that a packet supports ECN, ECT(0) and ECT(1), with the second codepoint used to support ECN nonce functionality to discourage receivers from exploiting ECN to improve their throughput at the expense of other network users, as specified in experimental RFC 3540 [RFC3540].

While the ECN Nonce works as specified, and has been deployed in limited environments, widespread usage in the Internet has not materialized. A study of the ECN behaviour of the Alexa top 1M web servers using 2014 data [Trammell15] found that after ECN was negotiated, none of the 581,711 IPv4 servers tested were using both ECT codepoints, which would have been a possible sign of ECN Nonce usage. Of the 17,028 IPv6 servers tested, 4 set both ECT(0) and ECT(1) on data packets. This might have been evidence of use of the ECN Nonce by these 4 servers, but equally it might have been due to re-marking of the ECN field by an erroneous middlebox or router.

With the emergence of new experimental functionality that depends on use of the ECT(1) codepoint for other purposes, continuing to reserve that codepoint for the ECN Nonce experiment is no longer justified. In addition, other approaches to discouraging receivers from exploiting ECN have emerged, see Appendix B.1 of [I-D.briscoe-tsvwg-ecn-l4s-id]. Therefore, in support of ECN experimentation with the ECT(1) codepoint, this memo:

- o Declares that the ECN Nonce experiment [RFC3540] has concluded, and notes the absence of widespread deployment.

- o Obsoletes RFC 3540 in order to facilitate experimental use of the ECT(1) codepoint.
- o Reclassifies RFC 3540 as Historic to document the ECN Nonce experiment and discourage further implementation of the ECN Nonce.
- o Updates RFC 3168 [RFC3168] to remove discussion of the ECN Nonce and use of ECT(1) for that Nonce. The specific text updates are omitted for brevity.

The following guidance on ECT codepoint usage in the ECN field of IP headers from Section 5 of RFC 3168 is relevant when the ECN Nonce is not implemented:

Protocols and senders that only require a single ECT codepoint SHOULD use ECT(0).

OPEN ISSUE: Change the above requirement in RFC 3168 from SHOULD to MUST towards reserving ECT(1) for experimentation?

4. Updates to RFC 3168

RFC 3168 is a Proposed Standard RFC, so updating RFC 3168 requires publishing a standards track RFC unless a standards process exception is approved by the IESG, e.g., to allow an Experimental RFC to update RFC 3168. In support of the above areas of experimentation, and specifically to avoid multiple uncoordinated requests to the IESG for standards process exceptions, this memo updates RFC 3168 [RFC3168] to allow changes in the following areas, provided that the changes are documented by an Experimental RFC. It is also possible to change RFC 3168 via publication of another standards track RFC.

4.1. Congestion Response Differences

Section 5 of RFC 3168 specifies 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.

In support of Congestion Response Differences experimentation, this memo updates RFC 3168 to allow the congestion control response (including the TCP Sender's congestion control response) to a CE-marked packet to differ from the response to a dropped packet, provided that the changes from RFC 3168 are documented in an Experimental RFC. The specific change to RFC 3168 is to insert the

words "unless otherwise specified by an Experimental RFC" at the end of the sentence quoted above.

RFC 4774 [RFC4774] quotes the above text from RFC 3168 as background, but does not impose requirements based on that text. Therefore no update to RFC 4774 is required to enable this area of experimentation.

4.2. ECT Differences

Section 5 of RFC 3168 specifies that:

Routers treat the ECT(0) and ECT(1) codepoints as equivalent.

In support of ECT Differences experimentation, this memo updates RFC 3168 to allow routers to treat the ECT(0) and ECT(1) codepoints differently, provided that the changes from RFC 3168 are documented in an Experimental RFC. The specific change to RFC 3168 is to insert the words "unless otherwise specified by an Experimental RFC" at the end of the above sentence.

In support of ECT Differences experimentation, this memo updates RFC 3168 to enable effective endpoint use of ECT(1) for large scale experimentation. The proposed L4S experiment [I-D.briscoe-tsvwg-ecn-l4s-id] significantly increases the CE marking probability for ECT(1)-marked traffic in a fashion that interacts badly with existing sender congestion response functionality because that functionality assumes that a CE-marked packet would have been dropped by the network. If network traffic that uses such a sender congestion response encounters L4S's increased marking probability (and hence rate) at a network bottleneck queue, the resulting traffic throughput is likely to be much less than intended for the level of congestion at the bottleneck queue. To avoid that interaction, this memo reserves ECT(1) for experimentation, initially L4S. The specific update to Section 5 of RFC 3168 is to remove the following text:

Senders are free to use either the ECT(0) or the ECT(1) codepoint to indicate ECT, on a packet-by-packet basis.

The use of both the two codepoints for ECT, ECT(0) and ECT(1), is motivated primarily by the desire to allow mechanisms for the data sender to verify that network elements are not erasing the CE codepoint, and that data receivers are properly reporting to the sender the receipt of packets with the CE codepoint set, as required by the transport protocol. Guidelines for the senders and receivers to differentiate between the ECT(0) and ECT(1) codepoints will be addressed in separate documents, for each

transport protocol. In particular, this document does not address mechanisms for TCP end- nodes to differentiate between the ECT(0) and ECT(1) codepoints. Protocols and senders that only require a single ECT codepoint SHOULD use ECT(0).

and replace it with:

Unless otherwise modified by an Experimental RFC, senders MUST use the ECT(0) codepoint to indicate ECT and MUST NOT use the ECT(1) codepoint to indicate ECT.

ECT Differences experiments SHOULD modify the network behavior for ECT(1)-marked traffic rather than ECT(0)-marked traffic if network behavior for only one ECT codepoint is modified. ECT Differences experiments MUST NOT modify the network behavior for ECT(0)-marked traffic in a fashion that requires changes to sender congestion response to obtain desired network behavior. If an ECT Differences experiment modifies the network behavior for ECT(1)-marked traffic, e.g., CE-marking behavior, in a fashion that requires changes to sender congestion response to obtain desired network behavior, then the Experimental RFC for that experiment MUST specify:

- o The sender congestion response to CE marking in the network, and
- o Router behavior changes, or the absence thereof, in forwarding CE-marked packets that are part of the experiment.

In addition, until the conclusion of the L4S experiment, use of ECT(1) in IETF RFCs is not appropriate, as the IETF may decide to allocate ECT(1) exclusively for L4S usage if the L4S experiment is successful.

In support of ECT Differences experimentation, this memo also updates RFC 3168 to remove discussion of the ECN Nonce, as noted in Section 3 above.

4.3. Generalized ECN

RFC 3168 prohibits use of ECN for TCP control packets and retransmitted packets in a number of places:

- o "To ensure the reliable delivery of the congestion indication of the CE codepoint, an ECT codepoint MUST NOT be set in a packet unless the loss of that packet in the network would be detected by the end nodes and interpreted as an indication of congestion." (Section 5.2)

- o "A host MUST NOT set ECT on SYN or SYN-ACK packets."
(Section 6.1.1)
- o "pure acknowledgement packets (e.g., packets that do not contain any accompanying data) MUST be sent with the not-ECT codepoint."
(Section 6.1.4)
- o "This document specifies ECN-capable TCP implementations MUST NOT set either ECT codepoint (ECT(0) or ECT(1)) in the IP header for retransmitted data packets, and that the TCP data receiver SHOULD ignore the ECN field on arriving data packets that are outside of the receiver's current window." (Section 6.1.5)
- o "the TCP data sender MUST NOT set either an ECT codepoint or the CWR bit on window probe packets." (Section 6.1.6)

In support of Generalized ECN experimentation, this memo updates RFC 3168 to allow the use of ECT codepoints on SYN and SYN-ACK packets, pure acknowledgement packets, window probe packets and retransmissions of packets that were originally sent with an ECT codepoint, provided that the changes from RFC 3168 are documented in an Experimental RFC. The specific change to RFC 3168 is to insert the words "unless otherwise specified by an Experimental RFC" at the end of each sentence quoted above.

In addition, beyond requiring TCP senders not to set ECT on TCP control packets and retransmitted packets, RFC 3168 is silent on whether it is appropriate for a network element, e.g. a firewall, to discard such a packet as invalid. For Generalized ECN experimentation to be useful, middleboxes ought not to do that, therefore RFC 3168 is updated by adding the following text to the end of Section 6.1.1.1 on Middlebox Issues:

Unless otherwise specified by an Experimental RFC, middleboxes SHOULD NOT discard TCP control packets and retransmitted TCP packets solely because the ECN field in the IP header does not contain Not-ECT.

4.4. Effective Congestion Control is Required

Congestion control remains an important aspect of the Internet architecture [RFC2914]. Any Experimental RFC that takes advantage of this memo's updates to RFC 3168 or RFC 6679 is required to discuss the congestion control implications of the experiment(s) in order to provide assurance that deployment of the experiment(s) does not pose a congestion-based threat to the operation of the Internet.

5. ECN for RTP Updates to RFC 6679

RFC 6679 [RFC6679] specifies use of ECN for RTP traffic; it allows use of both the ECT(0) and ECT(1) codepoints, and provides the following guidance on use of these codepoints in section 7.3.1 :

The sender SHOULD mark packets as ECT(0) unless the receiver expresses a preference for ECT(1) or for a random ECT value using the "ect" parameter in the "a=ecn-capable-rtp:" attribute.

The ECT Differences area of experimentation increases the potential consequences of using ECT(1) instead of ECT(0), and hence the above guidance is updated by adding the following two sentences:

Random ECT values MUST NOT be used, as that may expose RTP to differences in network treatment of traffic marked with ECT(1) and ECT(0) and differences in associated endpoint congestion responses, e.g., as proposed in [I-D.briscoe-tsvwg-ecn-l4s-id]. In addition, ECT(0) MUST be used unless otherwise specified in an Experimental RFC.

Section 7.3.3 of RFC 6679 specifies RTP's response to receipt of CE marked packets as being identical to the response to dropped packets:

The reception of RTP packets with ECN-CE marks in the IP header is a notification that congestion is being experienced. The default reaction on the reception of these ECN-CE-marked packets MUST be to provide the congestion control algorithm with a congestion notification that triggers the algorithm to react as if packet loss had occurred. There should be no difference in congestion response if ECN-CE marks or packet drops are detected.

In support of Congestion Response Differences experimentation, this memo updates this text in a fashion similar to RFC 3168 to allow the RTP congestion control response to a CE-marked packet to differ from the response to a dropped packet, provided that the changes from RFC 6679 are documented in an Experimental RFC. The specific change to RFC 6679 is to insert the words "Unless otherwise specified by an Experimental RFC" and reformat the last two sentences to be subject to that condition, i.e.:

The reception of RTP packets with ECN-CE marks in the IP header is a notification that congestion is being experienced. Unless otherwise specified by an Experimental RFC:

- * The default reaction on the reception of these ECN-CE-marked packets MUST be to provide the congestion control algorithm

with a congestion notification that triggers the algorithm to react as if packet loss had occurred.

- * There should be no difference in congestion response if ECN-CE marks or packet drops are detected.

The second sentence of the immediately following paragraph in RFC 6679 requires a related update:

Other reactions to ECN-CE may be specified in the future, following IETF Review. Detailed designs of such additional reactions MUST be specified in a Standards Track RFC and be reviewed to ensure they are safe for deployment under any restrictions specified.

The update is to change "Standards Track RFC" to "Standards Track RFC or Experimental RFC" for consistency with the first update.

6. ECN for DCCP Updates to RFCs 4341, 4342 and 5622

The specifications of the three DCCP Congestion Control IDs (CCIDs) 2 [RFC4341], 3 [RFC4342] and 4 [RFC5622] contain broadly the same wording as follows:

each DCCP-Data and DCCP-DataAck packet is sent as ECN Capable with either the ECT(0) or the ECT(1) codepoint set.

This memo updates these sentences in each of the three RFCs as follows:

each DCCP-Data and DCCP-DataAck packet is sent as ECN Capable. Unless otherwise specified by an Experimental RFC, such DCCP senders SHOULD set the ECT(0) codepoint.

In support of ECT Differences experimentation (as noted in Section 3), this memo also updates all three of these RFCs to remove discussion of the ECN Nonce. The specific text updates are omitted for brevity.

7. Acknowledgements

The content of this draft, including the specific portions of RFC 3168 that are updated draws heavily from [I-D.khademi-tsvwg-ecn-response], whose authors are gratefully acknowledged. The authors of the Internet Drafts describing the experiments have motivated the production of this memo - their interest in innovation is welcome and heartily acknowledged. Colin

Perkins suggested updating RFC 6679 on RTP and provided guidance on where to make the updates.

The draft has been improved as a result of comments from a number of reviewers, including Spencer Dawkins, Gorrry Fairhurst, Ingemar Johansson, Naeem Khademi, Mirja Kuehlewind and Michael Welzl. Bob Briscoe's thorough review of an early version of this draft resulted in numerous improvements including addition of the updates to the DCCP RFCs.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

As a process memo that makes no changes to existing protocols, there are no protocol security considerations.

However, effective congestion control is crucial to the continued operation of the Internet, and hence this memo places the responsibility for not breaking Internet congestion control on the experiments and the experimenters who propose them, as specified in Section 4.4.

Security considerations for the proposed experiments are discussed in the Internet-Drafts that propose them.

See Appendix B.1 of [I-D.briscoe-tsvwg-ecn-l4s-id] for discussion of alternatives to the ECN Nonce.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<http://www.rfc-editor.org/info/rfc2914>>.
- [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>>.

- [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>>.
- [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>>.
- [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>>.
- [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>>.

10.2. Informative References

- [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-02 (work in progress), October 2016.
- [I-D.khademi-tcpm-alternativebackoff-ecn]
Khademi, N., Welzl, M., Armitage, G., and G. Fairhurst, "TCP Alternative Backoff with ECN (ABE)", draft-khademi-tcpm-alternativebackoff-ecn-01 (work in progress), October 2016.

[I-D.khademi-tsvwg-ecn-response]

Khademi, N., Welzl, M., Armitage, G., and G. Fairhurst,
"Updating the Explicit Congestion Notification (ECN)
Specification to Allow IETF Experimentation", draft-
khademi-tsvwg-ecn-response-01 (work in progress), July
2016.

[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>>.

[Trammell15]

Trammell, B., Kuehlewind, M., Boppart, D., Learmonth, I.,
Fairhurst, G., and R. Scheffenegger, "Enabling Internet-
Wide Deployment of Explicit Congestion Notification".

In Proc Passive & Active Measurement (PAM'15) Conference
(2015)

Appendix A. Change History

[To be removed before RFC publication.]

Changes from draft-black-tsvwg-ecn-experimentation-00 to -01:

- o Section 4.2 - also update RFC 3168 to remove sentence indicating that senders are free to use both ECT codepoints. Add a SHOULD for ECT Differences experiments to use ECT(1).
- o Section 5 - only discourage use of random ECT values, but use NOT RECOMMENDED to do so. Consistent use of ECT(1) without using ECT(0) is ok. Mention possible changes in endpoint response.
- o Add more Acknowledgements and Change History
- o Additional editorial changes.

Changes from draft-black-tsvwg-ecn-experimentation-01 to -02:

- o Add DCCP RFC updates and one missing RFC 3168 update (probe packets).
- o Discourage RTP usage of ECT(1).
- o Strengthen text on lack of ECN Nonce deployment.

- o Cross-reference the L4S draft appendix that discusses ECN Nonce alternatives.
- o Additional editorial changes.

Changes from draft-black-tsvwg-ecn-experimentation-02 to -03:

- o Clarify that "SHOULD use ECT(0)" guidance from RFC 3168 is about IP headers.
- o Add a "SHOULD NOT" requirement that middleboxes not discard TCP control packets, etc. solely because they use ECN.
- o Switch to pre-5378 boilerplate, due to vintage of RFCs being updated.
- o Additional editorial changes.

Changes from draft-black-tsvwg-ecn-experimentation-03 to -04:

- o Use "Congestion Response Differences" as name of experimentation area instead of "Alternative Backoff" to avoid confusion with specific experiment.
- o Change ECT(1) requirement to "MUST NOT use unless otherwise specified by an Experimental RFC" This resulted in extensive changes to Section 4.2.
- o Clean up and tighten language requiring all congestion responses to be IETF-approved
- o Additional editorial changes.

Author's Address

David Black
Dell EMC
176 South Street
Hopkinton, MA 01748
USA

Email: david.black@dell.com

Active Queue Management (aqm)
Internet-Draft
Intended status: Experimental
Expires: May 4, 2017

K. De Schepper
Nokia Bell Labs
B. Briscoe, Ed.
O. Bondarenko
Simula Research Lab
I. Tsang
Nokia Bell Labs
October 31, 2016

DualQ Coupled AQM for Low Latency, Low Loss and Scalable Throughput
draft-briscoe-tsvwg-aqm-dualq-coupled-00

Abstract

Data Centre TCP (DCTCP) was designed to provide predictably low queuing latency, near-zero loss, and throughput scalability using explicit congestion notification (ECN) and an extremely simple marking behaviour on switches. However, DCTCP does not co-exist with existing TCP traffic---throughput starves. So, until now, DCTCP could only be deployed where a clean-slate environment could be arranged, such as in private data centres. This specification defines 'DualQ Coupled Active Queue Management (AQM)' to allow scalable congestion controls like DCTCP to safely co-exist with classic Internet traffic. The Coupled AQM ensures that a flow runs at about the same rate whether it uses DCTCP or TCP Reno/Cubic, but without inspecting transport layer flow identifiers. When tested in a residential broadband setting, DCTCP achieved sub-millisecond average queuing delay and zero congestion loss under a wide range of mixes of DCTCP and 'Classic' broadband Internet traffic, without compromising the performance of the Classic traffic. The solution also reduces network complexity and eliminates network configuration.

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 May 4, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Problem and Scope	2
1.2. Terminology	5
1.3. Features	5
2. DualQ Coupled AQM Algorithm	6
2.1. Coupled AQM	7
2.2. Dual Queue	8
2.3. Traffic Classification	8
2.4. Normative Requirements	8
3. IANA Considerations	9
4. Security Considerations	10
4.1. Overload Handling	10
5. Acknowledgements	11
6. References	11
6.1. Normative References	11
6.2. Informative References	12
Appendix A. Example DualQ Coupled PI2 Algorithm	14
Appendix B. Example DualQ Coupled Curvy RED Algorithm	17
Appendix C. Guidance on Controlling Throughput Equivalence	23
Authors' Addresses	24

1. Introduction

1.1. Problem and Scope

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 sawtooth 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.

It seems that further changes to the network alone will now yield diminishing returns. Data Centre TCP (DCTCP [I-D.ietf-tcpm-dctcp])

teaches us that a small but radical change to TCP is needed to cut two major outstanding causes of queuing delay variability:

1. the 'sawtooth' varying rate of TCP itself;
2. the smoothing delay deliberately introduced into AQMs to permit bursts without triggering losses.

The former causes a flow's round trip time (RTT) to vary from about 1 to 2 times the base RTT between the machines in question. The latter delays the system's response to change by a worst-case (transcontinental) RTT, which could be hundreds of times the actual RTT of typical traffic from localized CDNs.

Latency is not our only concern:

3. 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 cause 'classic' TCP to starve itself, which is why they have been confined to private data centres or research testbeds (until now).

This document specifies a 'DualQ Coupled AQM' extension that solves the problem of coexistence between scalable and classic flows, without having to inspect flow identifiers. The AQM is not like flow-queuing approaches [I-D.ietf-aqm-fq-codel] that classify packets by flow identifier into numerous separate queues in order to isolate sparse flows from the higher latency in the queues assigned to heavier flow. In contrast, the AQM exploits the behaviour of scalable congestion controls like DCTCP so that every packet in every flow sharing the queue for DCTCP-like traffic can be served with very low latency.

This AQM extension can be combined with any single queue AQM that generates a statistical or deterministic mark/drop probability driven by the queue dynamics. In many cases it simplifies the basic control algorithm, and requires little extra processing. Therefore it is believed the Coupled AQM would be applicable and easy to deploy in all types of buffers; buffers in cost-reduced mass-market residential equipment; buffers in end-system stacks; buffers in carrier-scale equipment including remote access servers, routers, firewalls and

Ethernet switches; buffers in network interface cards, buffers in virtualized network appliances, hypervisors, and so on.

The supporting papers [PI216] and [DCTtH15] give the full rationale for the AQM's design, both discursively and in more precise mathematical form.

1.2. Terminology

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

The DualQ Coupled AQM uses two queues for two services. Each of the following terms identifies both the service and the queue that provides the service:

Classic (denoted by subscript C): The 'Classic' service is intended for all the behaviours that currently co-exist with TCP Reno (TCP Cubic, Compound, SCTP, etc).

Low-Latency, Low-Loss and Scalable (L4S, denoted by subscript L): The 'L4S' service is intended for a set of congestion controls with scalable properties such as DCTCP (e.g. Relentless [Mathis09]).

Either service can cope with a proportion of unresponsive or less-responsive traffic as well (e.g. DNS, VoIP, etc), just as a single queue AQM can. The DualQ Coupled AQM behaviour is similar to a single FIFO queue with respect to unresponsive and overload traffic.

1.3. Features

The AQM couples marking and/or dropping across the two queues such that a flow will get roughly the same throughput whichever it uses. Therefore both queues can feed into the full capacity of a link and no rates need to be configured for the queues. The L4S queue enables scalable congestion controls like DCTCP to give stunningly low and predictably low latency, without compromising the performance of competing 'Classic' Internet traffic. Thousands of tests have been conducted in a typical fixed residential broadband setting. Typical experiments used base round trip delays up to 100ms between the data centre and home network, and large amounts of background traffic in both queues. For every L4S packet, the AQM kept the average queuing delay below 1ms (or 2 packets if serialization delay is bigger for

slow links), and no losses at all were introduced by the AQM. Details of the extensive experiments will be made available [PI216] [DCTtH15].

Subjective testing was also conducted using a demanding panoramic interactive video application run over a stack with DCTCP enabled and deployed on the testbed. Each user could pan or zoom their own high definition (HD) sub-window of a larger video scene from a football match. Even though the user was also downloading large amounts of L4S and Classic data, latency was so low that the picture appeared to stick to their finger on the touchpad (all the L4S data achieved the same ultra-low latency). With an alternative AQM, the video noticeably lagged behind the finger gestures.

Unlike Diffserv Expedited Forwarding, the L4S queue does not have to be limited to a small proportion of the link capacity in order to achieve low delay. The L4S queue can be filled with a heavy load of capacity-seeking flows like DCTCP and still achieve low delay. The L4S queue does not rely on the presence of other traffic in the Classic queue that can be 'overtaken'. It gives low latency to L4S traffic whether or not there is Classic traffic, and the latency of Classic traffic does not suffer when a proportion of the traffic is L4S. The two queues are only necessary because DCTCP-like flows cannot keep latency predictably low and keep utilization high if they are mixed with legacy TCP flows,

The experiments used the Linux implementation of DCTCP that is deployed in private data centres, without any modification despite its known deficiencies. Nonetheless, certain modifications will be necessary before DCTCP is safe to use on the Internet, which are recorded for now in Appendix A of [I-D.briscoe-tsvwg-aqm-tcpm-rmcat-l4s-problem]. However, the focus of this specification is to get the network service in place. Then, without any management intervention, applications can exploit it by migrating to scalable controls like DCTCP, which can then evolve while their benefits are being enjoyed by everyone on the Internet.

2. DualQ Coupled AQM Algorithm

There are two main aspects to the algorithm:

- o the Coupled AQM that addresses throughput equivalence between Classic (e.g. Reno, Cubic) flows and L4S (e.g. DCTCP) flows
- o the Dual Queue structure that provides latency separation for L4S flows to isolate them from the typically large Classic queue.

2.1. Coupled AQM

In the 1990s, the 'TCP formula' was derived for the relationship between TCP's congestion window, *cwnd*, and its drop probability, *p*. To a first order approximation, *cwnd* of TCP Reno is inversely proportional to the square root of *p*. TCP Cubic implements a Reno-compatibility mode, which is the only relevant mode for typical RTTs under 20ms, while the throughput of a single flow is less than about 500Mb/s. Therefore we can assume that Cubic traffic behaves similar to Reno (but with a slightly different constant of proportionality), and we shall use the term 'Classic' for the collection of Reno and Cubic in Reno mode.

In our supporting paper [PI216], we derive the equivalent rate equation for DCTCP, for which *cwnd* is inversely proportional to *p* (not the square root), where in this case *p* is the ECN marking probability. DCTCP is not the only congestion control that behaves like this, so we use the term 'L4S' traffic for all similar behaviour.

In order to make a DCTCP flow run at roughly the same rate as a Reno TCP flow (all other factors being equal), we make the drop or marking probability for Classic traffic, *p_C* distinct from the marking probability for L4S traffic, *p_L* (in contrast to RFC3168 which requires them to be the same). We make the Classic drop probability *p_C* proportional to the square of the L4S marking probability *p_L*. This is because we need to make the Reno flow rate equal the DCTCP flow rate, so we have to square the square root of *p_C* in the Reno rate equation to make it the same as the straight *p_L* in the DCTCP rate equation.

There is a really simple way to implement the square of a probability - by testing the queue against two random numbers not one. This is the approach adopted in Appendix A and Appendix B.

Stating this as a formula, the relation between Classic drop probability, *p_C*, and L4S marking probability, *p_L* needs to take the form:

$$p_C = (p_L / k)^2 \quad (1)$$

where *k* is the constant of proportionality. Optionally, *k* can be expressed as a power of 2, so *k*=2^{*k'*}, where *k'* is another constant. Then implementations can avoid costly division by shifting *p_L* by *k'* bits to the right.

2.2. Dual Queue

Classic traffic builds a large queue, so a separate queue is provided for L4S traffic, and it is scheduled with strict priority. Nonetheless, coupled marking ensures that giving priority to L4S traffic still leaves the right amount of spare scheduling time for Classic flows to each get equivalent throughput to DCTCP flows (all other factors such as RTT being equal). The algorithm achieves this without having to inspect flow identifiers.

2.3. Traffic Classification

Both the Coupled AQM and DualQ mechanisms need an identifier to distinguish L4S and C packets. A separate draft [I-D.briscoe-tsvwg-ecn-l4s-id] recommends using the ECT(1) codepoint of the ECN field as this identifier, having assessed various alternatives.

Given L4S work is currently on the experimental track, but the definition of the ECN field is on the standards track [RFC3168], another standards track document has proved necessary to make the ECT(1) codepoint available for experimentation [I-D.black-tsvwg-ecn-experimentation].

2.4. Normative Requirements

In the Dual Queue, L4S packets **MUST** be given priority over Classic, although strict priority **MAY** not be appropriate.

All L4S traffic **MUST** be ECN-capable, although some Classic traffic **MAY** also be ECN-capable.

Whatever identifier is used for L4S traffic, it will still be necessary to agree on the meaning of an ECN marking on L4S traffic, relative to a drop of Classic traffic. In order to prevent starvation of Classic traffic by scalable L4S traffic (e.g. DCTCP) the drop probability of Classic traffic **MUST** be proportional to the square of the marking probability of L4S traffic, In other words, the power to which p_L is raised in Eqn. (1) **MUST** be 2.

The constant of proportionality, k , in Eqn (1) determines the relative flow rates of Classic and L4S flows when the AQM concerned is the bottleneck (all other factors being equal). k does not have to be standardized because differences do not prevent interoperability. However, k has to take some value, and each operator can make that choice.

A value of $k=2$ is currently RECOMMENDED as the default for Internet access networks. Assuming scalable congestion controls for the Internet will be as aggressive as DCTCP, this will ensure their congestion window will be roughly the same as that of a standards track TCP congestion control (Reno) [RFC5681] and other so-called TCP-friendly controls such as TCP Cubic in its TCP-friendly mode.

The requirements for scalable congestion controls on the Internet (termed the TCP Prague requirements) are only in initial draft form [I-D.briscoe-tsvwg-aqm-tcpm-rmcat-l4s-problem] and subject to change. If the aggressiveness of DCTCP is not defined as the benchmark for scalable controls on the Internet, the recommended value of k will also be subject to change.

Whatever value is recommended, the choice of k is a matter of operator policy, and operators MAY choose a different value using Table 1 and the guidelines in Appendix C.

Typically, access network operators isolate customers from each other with some form of layer-2 multiplexing (TDM in DOCSIS, CDMA in 3G) or L3 scheduling (WRR in broadband), rather than relying on TCP to share capacity between customers [RFC0970]. In such cases, the choice of k will solely affect relative flow rates within each customer's access capacity, not between customers. Also, k will not affect relative flow rates at any times when all flows are Classic or all L4S, and it will not affect small flows.

Example DualQ Coupled AQM algorithms called PI2 and Curvy RED are given in Appendix A and Appendix B. Either example AQM can be used to couple packet marking and dropping across a dual Q. Curvy RED requires less operations per packet than RED and can be used if the range of RTTs is limited. PI2 is a simplification of PIE with stable Proportional-Integral control for both Classic and L4S congestion controls. Nonetheless, it would be possible to control the queues with other alternative AQMs, as long as the above normative requirements (those expressed in capitals) are observed, which are intended to be independent of the specific AQM.

{ToDo: Add management and monitoring requirements}

3. IANA Considerations

This specification contains no IANA considerations.

4. Security Considerations

4.1. Overload Handling

Where the interests of users or flows might conflict, it could be necessary to police traffic to isolate any harm to performance. This is a policy issue that needs to be separable from a basic AQM, but an AQM does need to handle overload. A trade-off needs to be made between complexity and the risk of either class harming the other. It is an operator policy to define what must happen if the service time of the classic queue becomes too great. In the following subsections three optional non-exclusive overload protections are defined. Their objective is for the overload behaviour of the DualQ AQM to be similar to a single queue AQM. The example implementation in Appendix A implements the 'delay on overload' policy. Other overload protections can be envisaged:

Minimum throughput service: By replacing the priority scheduler with a weighted round robin scheduler, a minimum throughput service can be guaranteed for Classic traffic. Typically the scheduling weight of the Classic queue will be small (e.g. 5%) to avoid interference with the coupling but big enough to avoid complete starvation of Classic traffic.

Delay on overload: To control milder overload of responsive traffic, particularly when close to the maximum congestion signal, delay can be used as an alternative congestion control mechanism. The Dual Queue Coupled AQM can be made to behave like a single First-In First-Out (FIFO) queue with different service times by replacing the priority scheduler with a very simple scheduler that could be called a "time-shifted FIFO", which is the same as the Modifier Earliest Deadline First (MEDF) scheduler of [MEDF]. The scheduler adds T_m to the queue delay of the next L4S packet, before comparing it with the queue delay of the next Classic packet, then it selects the packet with the greater adjusted queue delay. Under regular conditions, this time-shifted FIFO scheduler behaves just like a strict priority scheduler. But under moderate or high overload it prevents starvation of the Classic queue, because the time-shift defines the maximum extra queuing delay (T_m) of Classic packets relative to L4S.

Drop on overload: On severe overload, e.g. due to non responsive traffic, queues will typically overflow and packet drop will be unavoidable. It is important to avoid unresponsive ECN traffic (either Classic or L4S) driving the AQM to 100% drop and mark probability. Congestion controls that have a minimum congestion window will become unresponsive to ECN marking when the marking probability is high. This situation can be avoided by applying

the drop probability to all packets of all traffic types when it exceeds a certain threshold or by limiting the drop and marking probabilities to a lower maximum value (up to where fairness between the different traffic types is still guaranteed) and rely on delay to control temporary high congestion and eventually queue overflow. If the classic drop probability is applied to all types of traffic when it is higher than a threshold probability the queueing delay can be controlled up to any overload situation, and no further measures are required. If a maximum classic and coupled L4S probability of less than 100% is used, both queues need scheduling opportunities and should eventually experience drop. This can be achieved with a scheduler that guarantees a minimum throughput for each queue, such as a weighted round robin or time-shifted FIFO scheduler. In that case a common queue limit can be configured that will drop packets of both types of traffic.

To keep the throughput of both L4S and Classic flows equal over the full load range, a different control strategy needs to be defined above the point where one congestion control first saturates to a probability of 100% (if $k > 1$, L4S will saturate first). Possible strategies include: also dropping L4S; increasing the queueing delay for both; or ensuring that L4S traffic still responds to marking below a window of 2 segments (see Appendix A of [I-D.briscoe-tsvwg-aqm-tcpm-rmcat-l4s-problem]).

5. Acknowledgements

Thanks to Anil Agarwal for detailed review comments and suggestions on how to make our explanation clearer.

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

6. References

6.1. Normative References

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

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>>.
- [CoDel] Nichols, K. and V. Jacobson, "Controlling Queue Delay", ACM Queue 10(5), May 2012, <<http://queue.acm.org/issuedetail.cfm?issue=2208917>>.
- [CRED_Insights] Briscoe, B., "Insights from Curvy RED (Random Early Detection)", BT Technical Report TR-TUB8-2015-003, July 2015, <http://www.bobbriscoe.net/projects/latency/credi_tr.pdf>.
- [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.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-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.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-02 (work in progress), October 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-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.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.
- [Mathis09]
Mathis, M., "Relentless Congestion Control", PFLDNeT'09 , May 2009, <http://www.hpcc.jp/pfldnet2009/Program_files/1569198525.pdf>.
- [MEDF]
Menth, M., Schmid, M., Heiss, H., and T. Reim, "MEDF - a simple scheduling algorithm for two real-time transport service classes with application in the UTRAN", Proc. IEEE Conference on Computer Communications (INFOCOM'03) Vol.2 pp.1116-1122, March 2003.
- [PI216]
De Schepper, K., Bondarenko, O., Briscoe, B., and I. Tsang, "PI2: A Linearized AQM for both Classic and Scalable TCP", ACM CoNEXT'16 , December 2016, <https://riteproject.files.wordpress.com/2015/10/pi2_conext.pdf>.
- (To appear)
- [RFC0970]
Nagle, J., "On Packet Switches With Infinite Storage", RFC 970, DOI 10.17487/RFC0970, December 1985, <<http://www.rfc-editor.org/info/rfc970>>.

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

Appendix A. Example DualQ Coupled PI2 Algorithm

As a first concrete example, the pseudocode below gives the DualQ Coupled AQM algorithm based on the PI2 Classic AQM, we used and tested. For this example only the pseudo code is given. An open source implementation for Linux is available at: <https://github.com/olgabo/dualpi2>.

```
1: dualpi2_enqueue(lq, cq, pkt) { % Test limit and classify lq or cq
2:   stamp(pkt)                  % attach arrival time to packet
3:   if ( lq.len() + cq.len() > limit )
4:     drop(pkt)                  % drop packet if q is full
5:   else {
6:     if ( ecn(pkt) modulo 2 == 0 ) % ECN bits = not-ect or ect(0)
7:       cq.enqueue(pkt)
8:     else                        % ECN bits = ect(1) or ce
9:       lq.enqueue(pkt)
10:  }
11: }
```

Figure 1: Example Enqueue Pseudocode for DualQ Coupled PI2 AQM

```

1: dualpi2_dequeue(lq, cq) { % Couples L4S & Classic queues, lq & cq
2:   while ( lq.len() + cq.len() > 0 )
3:     if ( lq.time() + tshift >= cq.time() ) {
4:       lq.dequeue(pkt)
5:       if ( (pkt.time() > T) or (p > rand()) )
6:         mark(pkt)
7:       return(pkt) % return the packet and stop here
8:     } else {
9:       cq.dequeue(pkt)
10:      if ( p/k > max(rand(), rand()) ) % same as testing (p/k)^2
11:        if ( ecn(pkt) == 0 ) % ECN field = not-ect
12:          drop(pkt) % squared drop, redo loop
13:        else {
14:          mark(pkt) % squared mark
15:          return(pkt) % return the packet and stop here
16:        }
17:      else
18:        return(pkt) % return the packet and stop here
19:    }
20:  }
21:  return(NULL) % no packet to dequeue
22: }

```

Figure 2: Example Dequeue Pseudocode for DualQ Coupled PI2 AQM

```

1: dualpi2_update(lq, cq) { % Update p every Tupdate
2:   curq = cq.time() % use queuing time of first-in Classic packet
3:   alpha_U = alpha * Tupdate % done once when parameters are set
4:   beta_U = beta * Tupdate % done once when parameters are set
5:   p = p + alpha_U * (curq - target) + beta_U * (curq - prevq)
6:   prevq = curq
7: }

```

Figure 3: Example PI-Update Pseudocode for DualQ Coupled PI2 AQM

When packets arrive, first a common queue limit is checked as shown in line 3 of the enqueueing pseudocode in Figure 1. Note that the limit is deliberately tested before enqueue to avoid any bias against larger packets (so the actual buffer has to be one packet larger than limit). If limit is not exceeded, the packet will be classified and enqueued to the Classic or L4S queue dependent on the least significant bit of the ECN field in the IP header (line 6). Packets with a codepoint having an LSB of 0 (Not-ECT and ECT(0)) will be enqueued in the Classic queue. Otherwise, ECT(1) and CE packets will be enqueued in the L4S queue.

The pseudocode in Figure 2 summarises the per packet dequeue implementation of the DualPI2 code. Line 3 implements the time-

shifted FIFO scheduling. It takes the packet that waited the longest, biased by a time-shift of `tshift` for the Classic traffic. If an L4S packet is scheduled, lines 5 and 6 mark the packet if either the L4S threshold `T` is exceeded, or if a random marking decision is drawn according to the probability `p` (maintained by the `dualpi2_update()` function discussed below). If a Classic packet is scheduled, lines 10 to 16 drop or mark the packet based on 2 random decisions resulting in the squared probability $(p/k)^2$ (hence the name PI2 for Classic traffic). Note that `p` is reduced by the factor `k` here. This has 2 effects; first the steady state probability is halved as required to give Classic TCP and DCTCP traffic equal throughput; secondly, the effect of the dynamic gain parameters `alpha` and `beta` are halved as well, which is also needed give Classic TCP and DCTCP control the same stability.

The probability `p` is kept up to date by the core PI algorithm in Figure 3 which is executed every `Tupdate` ([I-D.ietf-aqm-pie] now recommends 16ms, but in our testing so far we have used the earlier recommendation of 32ms). Note that `p` solely depends on the queuing time in the Classic queue. In line 2, the current queuing delay is evaluated by inspecting the timestamp of the next packet to schedule in the Classic queue. The function `cq.time()` subtracts the time stamped at enqueue from the current time and implicitly takes the current queuing delay as 0 if the queue is empty. Line 3 and 4 only need to be executed when the configuration parameters are changed. `Alpha` and `beta` in Hz are gain factors per 1 second. If a briefer update time is configured, `alpha_U` and `beta_U` (`_U` = per `Tupdate`) also have to be reduced, to ensure that the same response is given over time. As such, a smaller `Tupdate` will only result in a response with smaller and finer steps, not a more aggressive response. The new probability is calculated in line 5, where `target` is the target queuing delay, as defined in [I-D.ietf-aqm-pie]. In corner cases, `p` can overflow the range `[0,1]` so the resulting value of `p` has to be bounded (omitted from the pseudocode). Unlike PIE, `alpha_U` and `beta_U` are not tuned dependent on `p`, every `Tupdate`. Instead, in PI2 `alpha_U` and `beta_U` can be constants because the squaring applied to Classic traffic tunes them inherently, as explained in [PI216].

In our experiments so far (building on experiments with PIE) on broadband access links ranging from 4 Mb/s to 200 Mb/s with base RTTs from 5 ms to 100 ms, PI2 achieves good results with the following parameters:

```
tshift = 40ms
```

```
T = max(1ms, serialization time of 2 MTU)
```

```
target = 20ms
```

Tupdate = 32ms

k = 2

alpha = 20Hz (alpha/k = 10Hz for Classic)

beta = 200Hz (beta/k = 100Hz for Classic)

Appendix B. Example DualQ Coupled Curvy RED Algorithm

As another example, the pseudocode below gives the Curvy RED based DualQ Coupled AQM algorithm we used and tested. Although we designed the AQM to be efficient in integer arithmetic, to aid understanding it is first given using real-number arithmetic. Then, one possible optimization for integer arithmetic is given, also in pseudocode. To aid comparison, the line numbers are kept in step between the two by using letter suffixes where the longer code needs extra lines.

```

1: dualq_dequeue(lq, cq) { % Couples L4S & Classic queues, lq & cq
2:   if ( lq.dequeue(pkt) ) {
3a:     p_L = cq.sec() / 2^S_L
3b:     if ( lq.bytt() > T )
3c:       mark(pkt)
3d:     elif ( p_L > maxrand(U) )
4:       mark(pkt)
5:     return(pkt) % return the packet and stop here
6:   }
7:   while ( cq.dequeue(pkt) ) {
8a:     alpha = 2^(-f_C)
8b:     Q_C = alpha * pkt.sec() + (1-alpha)* Q_C % Classic Q EWMA
9a:     sqrt_p_C = Q_C / 2^S_C
9b:     if ( sqrt_p_C > maxrand(2*U) )
10:       drop(pkt) % Squared drop, redo loop
11:     else
12:       return(pkt) % return the packet and stop here
13:   }
14:   return(NULL) % no packet to dequeue
15: }

16: maxrand(u) { % return the max of u random numbers
17:   maxr=0
18:   while (u-- > 0)
19:     maxr = max(maxr, rand()) % 0 <= rand() < 1
20:   return(maxr)
21: }
```

Figure 4: Example Dequeue Pseudocode for DualQ Coupled Curvy RED AQM

Packet classification code is not shown, as it is no different from Figure 1. Potential classification schemes are discussed in Section 2. Overload protection code will be included in a future draft {ToDo}.

At the outer level, the structure of `dualq_dequeue()` implements strict priority scheduling. The code is written assuming the AQM is applied on dequeue (Note 1). Every time `dualq_dequeue()` is called, the if-block in lines 2-6 determines whether there is an L4S packet to dequeue by calling `lq.dequeue(pkt)`, and otherwise the while-block in lines 7-13 determines whether there is a Classic packet to dequeue, by calling `cq.dequeue(pkt)`. (Note 2)

In the lower priority Classic queue, a while loop is used so that, if the AQM determines that a classic packet should be dropped, it continues to test for classic packets deciding whether to drop each until it actually forwards one. Thus, every call to `dualq_dequeue()` returns one packet if at least one is present in either queue, otherwise it returns NULL at line 14. (Note 3)

Within each queue, the decision whether to drop or mark is taken as follows (to simplify the explanation, it is assumed that $U=1$):

L4S: If the test at line 2 determines there is an L4S packet to dequeue, the tests at lines 3a and 3c determine whether to mark it. The first is a simple test of whether the L4S queue (`lq.bytt()` in bytes) is greater than a step threshold T in bytes (Note 4). The second test is similar to the random ECN marking in RED, but with the following differences: i) the marking function does not start with a plateau of zero marking until a minimum threshold, rather the marking probability starts to increase as soon as the queue is positive; ii) marking depends on queuing time, not bytes, in order to scale for any link rate without being reconfigured; iii) marking of the L4S queue does not depend on itself, it depends on the queuing time of the `_other_` (Classic) queue, where `cq.sec()` is the queuing time of the packet at the head of the Classic queue (zero if empty); iv) marking depends on the instantaneous queuing time (of the other Classic queue), not a smoothed average; v) the queue is compared with the maximum of U random numbers (but if $U=1$, this is the same as the single random number used in RED).

Specifically, in line 3a the marking probability p_L is set to the Classic queueing time `qc.sec()` in seconds divided by the L4S scaling parameter 2^{S_L} , which represents the queuing time (in seconds) at which marking probability would hit 100%. Then in line 3d (if $U=1$) the result is compared with a uniformly distributed random number between 0 and 1, which ensures that marking

probability will linearly increase with queueing time. The scaling parameter is expressed as a power of 2 so that division can be implemented as a right bit-shift (>>) in line 3 of the integer variant of the pseudocode (Figure 5).

Classic: If the test at line 7 determines that there is at least one Classic packet to dequeue, the test at line 9b determines whether to drop it. But before that, line 8b updates Q_C , which is an exponentially weighted moving average (Note 5) of the queueing time in the Classic queue, where `pkt.sec()` is the instantaneous queueing time of the current Classic packet and α is the EWMA constant for the classic queue. In line 8a, α is represented as an integer power of 2, so that in line 8 of the integer code the division needed to weight the moving average can be implemented by a right bit-shift (>> f_C).

Lines 9a and 9b implement the drop function. In line 9a the averaged queueing time Q_C is divided by the Classic scaling parameter 2^{S_C} , in the same way that queueing time was scaled for L4S marking. This scaled queueing time is given the variable name `sqrtp_C` because it will be squared to compute Classic drop probability, so before it is squared it is effectively the square root of the drop probability. The squaring is done by comparing it with the maximum out of two random numbers (assuming $U=1$). Comparing it with the maximum out of two is the same as the logical 'AND' of two tests, which ensures drop probability rises with the square of queueing time (Note 6). Again, the scaling parameter is expressed as a power of 2 so that division can be implemented as a right bit-shift in line 9 of the integer pseudocode.

The marking/dropping functions in each queue (lines 3 & 9) are two cases of a new generalization of RED called Curvy RED, motivated as follows. When we compared the performance of our AQM with `fq_CoDel` and PIE, we came to the conclusion that their goal of holding queueing delay to a fixed target is misguided [CRED_Insights]. As the number of flows increases, if the AQM does not allow TCP to increase queueing delay, it has to introduce abnormally high levels of loss. Then loss rather than queueing becomes the dominant cause of delay for short flows, due to timeouts and tail losses.

Curvy RED constrains delay with a softened target that allows some increase in delay as load increases. This is achieved by increasing drop probability on a convex curve relative to queue growth (the square curve in the Classic queue, if $U=1$). Like RED, the curve hugs the zero axis while the queue is shallow. Then, as load increases, it introduces a growing barrier to higher delay. But, unlike RED, it requires only one parameter, the scaling, not three. The disadvantage

of Curvy RED is that it is not adapted to a wide range of RTTs. Curvy RED can be used as is when the RTT range to support is limited otherwise an adaptation mechanism is required.

There follows a summary listing of the two parameters used for each of the two queues:

Classic:

S_C : The scaling factor of the dropping function scales Classic queueing times in the range $[0, 2^{(S_C)}]$ seconds into a dropping probability in the range $[0,1]$. To make division efficient, it is constrained to be an integer power of two;

f_C : To smooth the queueing time of the Classic queue and make multiplication efficient, we use a negative integer power of two for the dimensionless EWMA constant, which we define as $2^{(-f_C)}$.

L4S :

S_L (and k): As for the Classic queue, the scaling factor of the L4S marking function scales Classic queueing times in the range $[0, 2^{(S_L)}]$ seconds into a probability in the range $[0,1]$. Note that $S_L = S_C + k$, where k is the coupling between the queues (Section 2.1). So S_L and k count as only one parameter;

T : The queue size in bytes at which step threshold marking starts in the L4S queue.

{ToDo: These are the raw parameters used within the algorithm. A configuration front-end could accept more meaningful parameters and convert them into these raw parameters.}

From our experiments so far, recommended values for these parameters are: $S_C = -1$; $f_C = 5$; $T = 5 * MTU$ for the range of base RTTs typical on the public Internet. [CRED_Insights] explains why these parameters are applicable whatever rate link this AQM implementation is deployed on and how the parameters would need to be adjusted for a scenario with a different range of RTTs (e.g. a data centre) {ToDo incorporate a summary of that report into this draft}. The setting of k depends on policy (see Section 2.4 and Appendix C respectively for its recommended setting and guidance on alternatives).

There is also a cUrviness parameter, U , which is a small positive integer. It is likely to take the same hard-coded value for all implementations, once experiments have determined a good value. We

have solely used $U=1$ in our experiments so far, but results might be even better with $U=2$ or higher.

Note that the dropping function at line 9 calls `maxrand(2*U)`, which gives twice as much curviness as the call to `maxrand(U)` in the marking function at line 3. This is the trick that implements the square rule in equation (1) (Section 2.1). This is based on the fact that, given a number X from 1 to 6, the probability that two dice throws will both be less than X is the square of the probability that one throw will be less than X . So, when $U=1$, the L4S marking function is linear and the Classic dropping function is squared. If $U=2$, L4S would be a square function and Classic would be quartic. And so on.

The `maxrand(u)` function in lines 16-21 simply generates u random numbers and returns the maximum (Note 7). Typically, `maxrand(u)` could be run in parallel out of band. For instance, if $U=1$, the Classic queue would require the maximum of two random numbers. So, instead of calling `maxrand(2*U)` in-band, the maximum of every pair of values from a pseudorandom number generator could be generated out-of-band, and held in a buffer ready for the Classic queue to consume.

```

1: dualq_dequeue(lq, cq) { % Couples L4S & Classic queues, lq & cq
2:   if ( lq.dequeue(pkt) ) {
3:     if ((lq.bytt() > T) || ((cq.ns() >> (S_L-2)) > maxrand(U)))
4:       mark(pkt)
5:     return(pkt) % return the packet and stop here
6:   }
7:   while ( cq.dequeue(pkt) ) {
8:     Q_C += (pkt.ns() - Q_C) >> f_C % Classic Q EWMA
9:     if ( (Q_C >> (S_C-2)) > maxrand(2*U) )
10:      drop(pkt) % Squared drop, redo loop
11:     else
12:      return(pkt) % return the packet and stop here
13:   }
14:   return(NULL) % no packet to dequeue
15: }
```

Figure 5: Optimised Example Dequeue Pseudocode for Coupled DualQ AQM using Integer Arithmetic

Notes:

1. The drain rate of the queue can vary if it is scheduled relative to other queues, or to cater for fluctuations in a wireless medium. To auto-adjust to changes in drain rate, the queue must be measured in time, not bytes or packets [CoDel]. In our Linux implementation, it was easiest to measure queuing time at

dequeue. Queuing time can be estimated when a packet is enqueued by measuring the queue length in bytes and dividing by the recent drain rate.

2. An implementation has to use priority queueing, but it need not implement strict priority.
3. If packets can be enqueued while processing dequeue code, an implementer might prefer to place the while loop around both queues so that it goes back to test again whether any L4S packets arrived while it was dropping a Classic packet.
4. In order not to change too many factors at once, for now, we keep the marking function for DCTCP-only traffic as similar as possible to DCTCP. However, unlike DCTCP, all processing is at dequeue, so we determine whether to mark a packet at the head of the queue by the byte-length of the queue `_behind_` it. We plan to test whether using queuing time will work in all circumstances, and if we find that the step can cause oscillations, we will investigate replacing it with a steep random marking curve.
5. An EWMA is only one possible way to filter bursts; other more adaptive smoothing methods could be valid and it might be appropriate to decrease the EWMA faster than it increases.
6. In practice at line 10 the Classic queue would probably test for ECN capability on the packet to determine whether to drop or mark the packet. However, for brevity such detail is omitted. All packets classified into the L4S queue have to be ECN-capable, so no dropping logic is necessary at line 3. Nonetheless, L4S packets could be dropped by overload code (see Section 4.1).
7. In the integer variant of the pseudocode (Figure 5) real numbers are all represented as integers scaled up by 2^{32} . In lines 3 & 9 the function `maxrand()` is arranged to return an integer in the range $0 \leq \text{maxrand}() < 2^{32}$. Queuing times are also scaled up by 2^{32} , but in two stages: i) In lines 3 and 8 queuing times `cq.ns()` and `pkt.ns()` are returned in integer nanoseconds, making the values about 2^{30} times larger than when the units were seconds, ii) then in lines 3 and 9 an adjustment of -2 to the right bit-shift multiplies the result by 2^2 , to complete the scaling by 2^{32} .

Appendix C. Guidance on Controlling Throughput Equivalence

RTT_C / RTT_L	Reno	Cubic
1	k=1	k=0
2	k=2	k=1
3	k=2	k=2
4	k=3	k=2
5	k=3	k=3

Table 1: Value of k for which DCTCP throughput is roughly the same as Reno or Cubic, for some example RTT ratios

To determine the appropriate policy, the operator first has to judge whether it wants DCTCP flows to have roughly equal throughput with Reno or with Cubic (because, even in its Reno-compatibility mode, Cubic is about 1.4 times more aggressive than Reno). Then the operator needs to decide at what ratio of RTTs it wants DCTCP and Classic flows to have roughly equal throughput. For example choosing the recommended value of k=0 will make DCTCP throughput roughly the same as Cubic, if their RTTs are the same.

However, even if the base RTTs are the same, the actual RTTs are unlikely to be the same, because Classic (Cubic or Reno) traffic needs a large queue to avoid under-utilization and excess drop, whereas L4S (DCTCP) does not. The operator might still choose this policy if it judges that DCTCP throughput should be rewarded for keeping its own queue short.

On the other hand, the operator will choose one of the higher values for k, if it wants to slow DCTCP down to roughly the same throughput as Classic flows, to compensate for Classic flows slowing themselves down by causing themselves extra queuing delay.

The values for k in the table are derived from the formulae, which was developed in [DCttH15]:

$$2^k = 1.64 (RTT_{reno} / RTT_{dc}) \quad (2)$$

$$2^k = 1.19 (RTT_{cubic} / RTT_{dc}) \quad (3)$$

For localized traffic from a particular ISP's data centre, we used the measured RTTs to calculate that a value of k=3 would achieve throughput equivalence, and our experiments verified the formula very closely.

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/>

Olga Bondarenko
Simula Research Lab
Lysaker
Norway

Email: olgabnd@gmail.com
URI: <https://www.simula.no/people/olgabo>

Ing-jyh Tsang
Nokia Bell Labs
Antwerp
Belgium

Email: ing-jyh.tsang@nokia.com

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.

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 May 4, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Problem	4
1.2. Terminology	5
1.3. Scope	6
2. L4S Packet Identifier	6
2.1. L4S Packet Identification Requirements	6
2.2. L4S Packet Identification	7
2.3. Pre-Requisite Transport Layer Behaviour	8
2.4. L4S Packet Identification by Network Nodes with Transport-Layer Awareness	9
2.5. The Meaning of CE Relative to Drop	10
3. IANA Considerations	10
4. Security Considerations	10
5. Acknowledgements	10
6. References	11
6.1. Normative References	11
6.2. Informative References	11
Appendix A. Alternative Identifiers	15
A.1. ECT(1) and CE codepoints	16
A.2. ECN Plus a Diffserv Codepoint (DSCP)	18
A.3. ECN capability alone	20
A.4. Protocol ID	21
A.5. Source or destination addressing	21
A.6. Summary: Merits of Alternative Identifiers	22

Appendix B. Potential Competing Uses for the ECT(1) Codepoint	23
B.1. Integrity of Congestion Feedback	23
B.2. Notification of Less Severe Congestion than CE	24
Authors' Addresses	24

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 sawtooth 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:

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

- o it SHOULD survive end-to-end between source and destination applications: across the boundary between host and network, between interconnected networks, and through middleboxes;
- o it SHOULD be common to IPv4 and IPv6 and transport agnostic;
- o it SHOULD be incrementally deployable;

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

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 mimic 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-sctp-ecn] 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

$$p_C \sim (p_L / k)^2$$

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 identifier 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](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](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 transport-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 signify 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 will 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 negotiation 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:

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

A.5. Source or destination addressing

Locally, a network operator could arrange for L4S service to be applied based on source or destination addressing, e.g. packets from its own data centre and/or CDN hosts, packets to its business

customers, etc. It could use addressing at any layer, e.g. IP addresses, MAC addresses, VLAN IDs, etc. Although addressing might be a useful tactical approach for a single ISP, it would not be a feasible approach to identify an end-to-end service like L4S. Even for a single ISP, it would require packet classifiers in buffers to be dependent on changing topology and address allocation decisions elsewhere in the network. Therefore this approach is not a feasible solution.

A.6. Summary: Merits of Alternative Identifiers

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

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

Note 1: Only feasible if classic ECN is obsolete.

Table 1: Comparison of the Merits of Three Alternative Identifiers

The schemes are scored based on both their capabilities now ('initial') and in the long term ('eventual'). The 'ECN' scheme shares the 'eventual' scores of the 'ECT(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:

- o the sender can test the integrity of the receiver's feedback by occasionally setting the IP-ECN field to a value normally only set by the network. Then it can test whether the receiver's feedback faithfully reports what it expects [I-D.moncaster-tcpm-rcv-cheat]. This works for loss and it will work for the accurate ECN feedback [RFC7560] intended for L4S;
- o 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 mechanism 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 identifier 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
Intended status: Informational
Expires: May 4, 2017

B. Briscoe, Ed.
Simula Research Lab
K. De Schepper
Nokia Bell Labs
M. Bagnulo Braun
Universidad Carlos III de Madrid
October 31, 2016

Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service:
Architecture
draft-briscoe-tsvwg-l4s-arch-00

Abstract

This document describes the L4S architecture for the provision of a new service that the Internet could provide to eventually replace best efforts for all traffic: Low Latency, Low Loss, Scalable throughput (L4S). It is becoming common for all (or most) applications being run by a user at any one time to require low latency. However, the only solution the IETF can offer for ultra-low queuing delay is Diffserv, which only favours a minority of packets at the expense of others. In extensive testing the new L4S service keeps average queuing delay under a millisecond for all applications even under very heavy load, without sacrificing utilization; and it keeps congestion loss to zero. It is becoming widely recognized that adding more access capacity gives diminishing returns, because latency is becoming the critical problem. Even with a high capacity broadband access, the reduced latency of L4S remarkably and consistently improves performance under load for applications such as interactive video, conversational video, voice, Web, gaming, instant messaging, remote desktop and cloud-based apps (even when all being used at once over the same access link). The insight is that the root cause of queuing delay is in TCP, not in the queue. By fixing the sending TCP (and other transports) queuing latency becomes so much better than today that operators will want to deploy the network part of L4S to enable new products and services. Further, the network part is simple to deploy - incrementally with zero-config. Both parts, sender and network, ensure coexistence with other legacy traffic. At the same time L4S solves the long-recognized problem with the future scalability of TCP throughput.

This document describes the L4S architecture, briefly describing the different components and how the work together to provide the aforementioned enhanced Internet service.

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 May 4, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. L4S architecture overview	4
3. Terminology	5
4. L4S architecture components	7
5. Rationale	8
5.1. Why These Primary Components?	8
5.2. Why Not Alternative Approaches?	10
6. Applicability statement	11
6.1. Use Cases	12
7. IANA Considerations	13
8. Security Considerations	13
8.1. Traffic (Non-)Policing	13
8.2. 'Latency Friendliness'	14
8.3. ECN Integrity	14

9. Acknowledgements	15
10. References	15
10.1. Normative References	15
10.2. Informative References	16
Appendix A. Required features for scalable transport protocols to be safely deployable in the Internet (a.k.a. TCP Prague requirements)	19
Appendix B. Standardization items	23
Authors' Addresses	25

1. Introduction

It is increasingly common for all of a user's applications at any one time to require low delay: interactive Web, Web services, voice, conversational video, interactive video, instant messaging, online gaming, remote desktop and cloud-based applications. In the last decade or so, much has been done to reduce propagation delay by placing caches or servers closer to users. However, queuing remains a major, albeit intermittent, component of latency. When present it typically doubles the path delay from that due to the base speed-of-light. Low loss is also important because, for interactive applications, losses translate into even longer retransmission delays.

It has been demonstrated that, once access network bit rates reach levels now common in the developed world, increasing capacity offers diminishing returns if latency (delay) is not addressed. Differentiated services (Diffserv) offers Expedited Forwarding [RFC3246] for some packets at the expense of others, but this is not applicable when all (or most) of a user's applications require low latency.

Therefore, the goal is an Internet service with ultra-Low queueing Latency, ultra-Low Loss and Scalable throughput (L4S) - for all traffic. This document describes the L4S architecture for achieving that goal.

It must be said that queuing delay only degrades performance infrequently [Hohlfeld14]. It only occurs when a large enough capacity-seeking (e.g. TCP) flow is running alongside the user's traffic in the bottleneck link, which is typically in the access network. Or when the low latency application is itself a large capacity-seeking flow (e.g. interactive video). At these times, the performance improvement must be so remarkable that network operators will be motivated to deploy it.

Active Queue Management (AQM) is part of the solution to queuing under load. AQM improves performance for all traffic, but there is a

limit to how much queuing delay can be reduced by solely changing the network; without addressing the root of the problem.

The root of the problem is the presence of standard TCP congestion control (Reno [RFC5681]) or compatible variants (e.g. TCP Cubic [I-D.ietf-tcpm-cubic]). We shall call this family of congestion controls 'Classic' TCP. It has been demonstrated that if the sending host replaces Classic TCP with a 'Scalable' alternative, when a suitable AQM is deployed in the network the performance under load of all the above interactive applications can be stunningly improved. For instance, queuing delay under heavy load with the example DCTCP/DualQ solution cited below is roughly 1 millisecond (1 ms) at the 99th percentile without losing link utilization. This compares with 5 to 20 ms on average with a Classic TCP and current state-of-the-art AQMs such as fq_CoDel [I-D.ietf-aqm-fq-codel] or PIE [I-D.ietf-aqm-pie]. Also, with a Classic TCP, 5 ms of queuing is usually only possible by losing some utilization.

It has been convincingly demonstrated [DCTtH15] that it is possible to deploy such an L4S service alongside the existing best efforts service so that all of a user's applications can shift to it when their stack is updated. Access networks are typically designed with one link as the bottleneck for each site (which might be a home, small enterprise or mobile device), so deployment at a single node should give nearly all the benefit. The L4S approach requires a number of mechanisms in different parts of the Internet to fulfill its goal. This document presents the L4S architecture, by describing the different components and how they interact to provide the scalable low-latency, low-loss, Internet service.

2. L4S architecture overview

There are three main components to the L4S architecture (illustrated in Figure 1):

- 2) Network: The L4S service traffic needs to be isolated from the queuing latency of the Classic service traffic. However, the two should be able to freely share a common pool of capacity. This is because there is no way to predict how many flows at any one time might use each service and capacity in access networks is too scarce to partition into two. So a 'semi-permeable' membrane is needed that partitions latency but not bandwidth. The Dual Queue Coupled AQM [I-D.briscoe-aqm-dualq-coupled] is an example of such a semi-permeable membrane.

Per-flow queuing such as in [I-D.ietf-aqm-fq-codel] could be used, but it partitions both latency and bandwidth between every end-to-end flow. So it is rather overkill, which brings disadvantages

(see Section 5.2), not least that thousands of queues are needed when two are sufficient.

- 1) Protocol: A host needs to distinguish L4S and Classic packets with an identifier so that the network can classify them into their separate treatments. [I-D.briscoe-tsvwg-ecn-l4s-id] considers various alternative identifiers, and concludes that all alternatives involve compromises, but the ECT(1) codepoint of the ECN field is a workable solution.
- 3) Host: Scalable congestion controls already exist. They solve the scaling problem with TCP first pointed out in [RFC3649]. The one used most widely (in controlled environments) is Data Centre TCP (DCTCP [I-D.ietf-tcpm-dctcp]), which has been implemented and deployed in Windows Server Editions (since 2012), in Linux and in FreeBSD. Although DCTCP as-is 'works' well over the public Internet, most implementations lack certain safety features that will be necessary once it is used outside controlled environments like data centres (see later). A similar scalable congestion control will also need to be transplanted into protocols other than TCP (SCTP, RTP/RTCP, RMCAT, etc.)

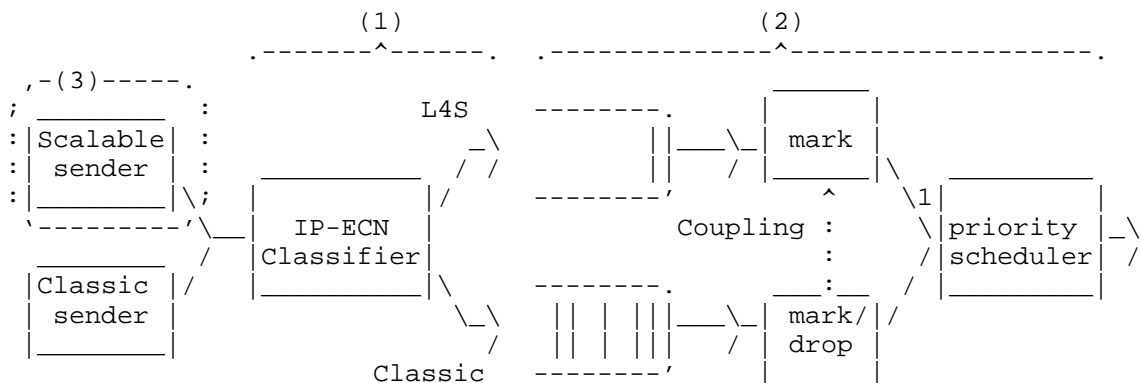


Figure 1: Components of an L4S Solution: 1) Isolation in separate network queues; 2) Packet Identification Protocol; and 3) Scalable Sending Host

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 [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. COMMENT: Since this will be an information document, This should be removed.

Classic service: The 'Classic' service is intended for all the congestion control 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).

Scalable Congestion Control: A congestion control where flow rate is inversely proportional to the level of congestion signals. Then, as flow rate scales, the number of congestion signals per round trip remains invariant, maintaining the same degree of control. For instance, DCTCP averages 2 congestion signals per round-trip whatever the flow rate.

Classic Congestion Control: A congestion control with a flow rate compatible with standard TCP Reno [RFC5681]. With Classic congestion controls, as capacity increases enabling higher flow rates, the number of round trips between congestion signals (losses or ECN marks) rises in proportion to the flow rate. So control of queuing and/or utilization becomes very slack. For instance, with 1500 B packets and an RTT of 18 ms, as TCP Reno flow rate increases from 2 to 100 Mb/s the number of round trips between congestion signals rises proportionately, from 2 to 100.

The default congestion control in Linux (TCP Cubic) is Reno-compatible for most scenarios expected for some years. For instance, with a typical domestic round-trip time (RTT) of 18ms, TCP Cubic only switches out of Reno-compatibility mode once the flow rate approaches 1 Gb/s. For a typical data centre RTT of 1 ms, the switch-over point is theoretically 1.3 Tb/s. However, with a less common transcontinental RTT of 100 ms, it only remains Reno-compatible up to 13 Mb/s. All examples assume 1,500 B packets.

Classic ECN: The original proposed standard Explicit Congestion Notification (ECN) protocol [RFC3168], which requires ECN signals to be treated the same as drops, both when generated in the network and when responded to by the sender.

Site: A home, mobile device, small enterprise or campus, where the network bottleneck is typically the access link to the site. Not all network arrangements fit this model but it is a useful, widely applicable generalisation.

4. L4S architecture components

The L4S architecture is composed by the following elements.

Protocols: The L4S architecture encompass the two protocols we describe next:

- a. [I-D.briscoe-tsvwg-ecn-l4s-id] recommends ECT(1) is used as the identifier to classify L4S and Classic packets into their separate treatments, as required by [RFC4774]. The draft also points out that the original experimental assignment of this codepoint as an ECN nonce [RFC3540] needs to be made obsolete (it was never deployed, and it offers no security benefit now that deployment is optional).
- b. An essential aspect of a scalable congestion control is the use of explicit congestion signals rather than losses, because the signals need to be sent immediately and frequently--too often to use drops. 'Classic' ECN [RFC3168] requires an ECN signal to be treated the same as a drop, both when it is generated in the network and when it is responded to by hosts. L4S allows networks and hosts to support two separate meanings for ECN. So the standards track [RFC3168] will need to be updated to allow ECT(1) packets to depart from the 'same as drop' constraint.

Network components: The Dual Queue Coupled AQM has been specified as generically as possible [I-D.briscoe-aqm-dualq-coupled] as a 'semi-permeable' membrane without specifying the particular AQMs to use in the two queues. An informational appendix of the draft is provided for pseudocode examples of different possible AQM approaches. Initially a zero-config variant of RED called Curvy RED was implemented, tested and documented. A variant of PIE has been implemented and tested and is about to be documented. The aim is for designers to be free to implement diverse ideas. So the brief normative body of the draft only specifies the minimum constraints an AQM needs to comply with to ensure that the L4S and Classic services will coexist.

Host mechanisms: The L4S architecture includes a number of mechanisms in the end host that we enumerate next:

- a. Data Centre TCP is the most widely used example of a scalable congestion control. It is being documented in the TCPM WG as an

informational record of the protocol currently in use [I-D.ietf-tcpm-dctcp]. It will be necessary to define a number of safety features for a variant usable on the public Internet. A draft list of these, known as the TCP Prague requirements, has been drawn up (see Appendix A).

- b. Transport protocols other than TCP use various congestion controls designed to be friendly with Classic TCP. It will be necessary to implement scalable variants of each of these transport behaviours before they can use the L4S service. The following standards track RFCs currently define these protocols, and they will need to be updated to allow a different congestion response, which they will have to indicate by using the ECT(1) codepoint: ECN in TCP [RFC3168], in SCTP [RFC4960], in RTP [RFC6679], and in DCCP [RFC4340].
- c. ECN feedback is sufficient for L4S in some transport protocols (RTCP, DCCP) but not others:
 - * For the case of TCP, the feedback protocol for ECN embeds the assumption from Classic ECN that it is the same as drop, making it unusable for a scalable TCP. Therefore, the implementation of TCP receivers will have to be upgraded [RFC7560]. Work to standardize more accurate ECN feedback for TCP (AccECN [I-D.ietf-tcpm-accurate-ecn]) is already in progress.
 - * ECN feedback is only roughly sketched in an appendix of the SCTP specification. A fuller specification has been proposed [I-D.stewart-tsvwg-sctpecn], which would need to be implemented and deployed.

5. Rationale

5.1. Why These Primary Components?

Explicit congestion signalling (protocol): Explicit congestion signalling is a key part of the L4S approach. In contrast, use of drop as a congestion signal creates a tension because drop is both a useful signal (more would reduce delay) and an impairment (less would reduce delay). Explicit congestion signals can be used many times per round trip, to keep tight control, without any impairment. Under heavy load, even more explicit signals can be applied so the queue can be kept short whatever the load. Whereas state-of-the-art AQMs have to introduce very high packet drop at high load to keep the queue short. Further, TCP's sawtooth reduction can be smaller, and therefore return to the operating point more often, without worrying that this causes more signals

(one at the top of each smaller sawtooth). The consequent smaller amplitude sawteeth fit between a very shallow marking threshold and an empty queue, so delay variation can be very low, without risk of under-utilization.

All the above makes it clear that explicit congestion signalling is only advantageous for latency if it does not have to be considered 'the same as' drop (as required with Classic ECN [RFC3168]). Before Classic ECN was standardized, there were various proposals to give an ECN mark a different meaning from drop. However, there was no particular reason to agree on any one of the alternative meanings, so 'the same as drop' was the only compromise that could be reached. RFC 3168 contains a statement that:

"An environment where all end nodes were ECN-Capable could allow new criteria to be developed for setting the CE codepoint, and new congestion control mechanisms for end-node reaction to CE packets. However, this is a research issue, and as such is not addressed in this document."

Latency isolation with coupled congestion notification (network):

Using just two queues is not essential to L4S (more would be possible), but it is the simplest way to isolate all the L4S traffic that keeps latency low from all the legacy Classic traffic that does not.

Similarly, coupling the congestion notification between the queues is not necessarily essential, but it is a clever and simple way to allow senders to determine their rate, packet-by-packet, rather than be overridden by a network scheduler. Because otherwise a network scheduler would have to inspect at least transport layer headers, and it would have to continually assign a rate to each flow without any easy way to understand application intent.

L4S packet identifier (protocol): Once there are at least two separate treatments in the network, hosts need an identifier at the IP layer to distinguish which treatment they intend to use.

Scalable congestion notification (host): A scalable congestion control keeps the signalling frequency high so that rate variations can be small when signalling is stable, and rate can track variations in available capacity as rapidly as possible otherwise.

5.2. Why Not Alternative Approaches?

All the following approaches address some part of the same problem space as L4S. In each case, it is shown that L4S complements them or improves on them, rather than being a mutually exclusive alternative:

Diffserv: Diffserv addresses the problem of bandwidth apportionment for important traffic as well as queuing latency for delay-sensitive traffic. L4S solely addresses the problem of queuing latency. Diffserv will still be necessary where important traffic requires priority (e.g. for commercial reasons, or for protection of critical infrastructure traffic). Nonetheless, if there are Diffserv classes for important traffic, the L4S approach can provide low latency for all traffic within each Diffserv class (including the case where there is only one Diffserv class).

Also, as already explained, Diffserv only works for a small subset of the traffic on a link. It is not applicable when all the applications in use at one time at a single site (home, small business or mobile device) require low latency. Also, because L4S is for all traffic, it needs none of the management baggage (traffic policing, traffic contracts) associated with favouring some packets over others. This baggage has held Diffserv back from widespread end-to-end deployment.

State-of-the-art AQMs: AQMs such as PIE and fq_CoDel give a significant reduction in queuing delay relative to no AQM at all. The L4S work is intended to complement these AQMs, and we definitely do not want to distract from the need to deploy them as widely as possible. Nonetheless, without addressing the large saw-toothed rate variations of Classic congestion controls, AQMs alone cannot reduce queuing delay too far without significantly reducing link utilization. The L4S approach resolves this tension by ensuring hosts can minimize the size of their sawteeth without appearing so aggressive to legacy flows that they starve.

Per-flow queuing: Similarly per-flow queuing is not incompatible with the L4S approach. However, one queue for every flow can be thought of as overkill compared to the minimum of two queues for all traffic needed for the L4S approach. The overkill of per-flow queuing has side-effects:

- A. fq makes high performance networking equipment costly (processing and memory) - in contrast dual queue code can be very simple;

- B. fq requires packet inspection into the end-to-end transport layer, which doesn't sit well alongside encryption for privacy - in contrast a dual queue only operates at the IP layer;
- C. fq decides packet-by-packet which flow to schedule without knowing application intent. In contrast, in the L4S approach the sender still controls the relative rate of each flow dependent on the needs of each application.

Alternative Back-off ECN (ABE): Yet again, L4S is not an alternative to ABE but a complement that introduces much lower queuing delay. ABE [I-D.khademi-tcpm-alternativebackoff-ecn] alters the host behaviour in response to ECN marking to utilize a link better and give ECN flows a faster throughput, but it assumes the network still treats ECN and drop the same. Therefore ABE exploits any lower queuing delay that AQMs can provide. But as explained above, AQMs still cannot reduce queuing delay too far without losing link utilization (for other non-ABE flows).

6. Applicability statement

A transport layer that solves the current latency issues will provide new service, product and application opportunities.

With the L4S approach, the following existing applications will immediately experience significantly better quality of experience under load in the best effort class:

- o Gaming
- o VoIP
- o Video conferencing
- o Web browsing
- o (Adaptive) video streaming
- o Instant messaging

The significantly lower queuing latency also enables some interactive application functions to be offloaded to the cloud that would hardly even be usable today:

- o Cloud based interactive video
- o Cloud based virtual and augmented reality

The above two applications have been successfully demonstrated with L4S, both running together over a 40 Mb/s broadband access link loaded up with the numerous other latency sensitive applications in the previous list as well as numerous downloads. A panoramic video of a football stadium can be swiped and pinched so that on the fly a proxy in the cloud generates a sub-window of the match video under the finger-gesture control of each user. At the same time, a virtual reality headset fed from a 360 degree camera in a racing car has been demonstrated, where the user's head movements control the scene generated in the cloud. In both cases, with 7 ms end-to-end base delay, the additional queuing delay of roughly 1 ms is so low that it seems the video is generated locally. See <https://riteproject.eu/dctth/> for videos of these demonstrations.

Using a swiping finger gesture or head movement to pan a video are extremely demanding applications--far more demanding than VoIP. Because human vision can detect extremely low delays of the order of single milliseconds when delay is translated into a visual lag between a video and a reference point (the finger or the orientation of the head).

If low network delay is not available, all fine interaction has to be done locally and therefore much more redundant data has to be downloaded. When all interactive processing can be done in the cloud, only the data to be rendered for the end user needs to be sent. Whereas, once applications can rely on minimal queues in the network, they can focus on reducing their own latency by only minimizing the application send queue.

6.1. Use Cases

The following use-cases for L4S are being considered by various interested parties:

- o Where the bottleneck is one of various types of access network:
 - DSL, cable, mobile, satellite
- * Radio links (cellular, WiFi) that are distant from the source are particularly challenging. The radio link capacity can vary rapidly by orders of magnitude, so it is often desirable to hold a buffer to utilise sudden increases of capacity;
- * cellular networks are further complicated by a perceived need to buffer in order to make hand-overs imperceptible;
- * Satellite networks generally have a very large base RTT, so even with minimal queuing, overall delay can never be extremely low;

- * Nonetheless, it is certainly desirable not to hold a buffer purely because of the sawteeth of Classic TCP, when it is more than is needed for all the above reasons.
- o Private networks of heterogeneous data centres, where there is no single administrator that can arrange for all the simultaneous changes to senders, receivers and network needed to deploy DCTCP:
 - * a set of private data centres interconnected over a wide area with separate administrations, but within the same company
 - * a set of data centres operated by separate companies interconnected by a community of interest network (e.g. for the finance sector)
 - * multi-tenant (cloud) data centres where tenants choose their operating system stack (Infrastructure as a Service - IaaS)
- o Different types of transport (or application) congestion control:
 - * elastic (TCP/SCTP);
 - * real-time (RTP, RMCAT);
 - * query (DNS/LDAP).
- o Where low delay quality of service is required, but without inspecting or intervening above the IP layer [I-D.you-encrypted-traffic-management]:
 - * mobile and other networks have tended to inspect higher layers in order to guess application QoS requirements. However, with growing demand for support of privacy and encryption, L4S offers an alternative. There is no need to select which traffic to favour for queuing, when L4S gives favourable queuing to all traffic.

7. IANA Considerations

This specification contains no IANA considerations.

8. Security Considerations

8.1. Traffic (Non-)Policing

Because the L4S service can serve all traffic that is using the capacity of a link, it should not be necessary to police access to the L4S service. In contrast, Diffserv only works if some packets

get less favourable treatment than others. So it has to use traffic policers to limit how much traffic can be favoured. In turn, traffic policers require traffic contracts between users and networks as well as pairwise between networks. Because L4S will lack all this management complexity, it is more likely to work end-to-end.

During early deployment (and perhaps always), some networks will not offer the L4S service. These networks do not need to police or remark L4S traffic - they just forward it unchanged as best efforts traffic, as they would already forward traffic with ECT(1) today. At a bottleneck, such networks will introduce some queuing and dropping. When a scalable congestion control detects a drop it will have to respond as if it is a Classic congestion control (see item 3-1 in Appendix A). This will ensure safe interworking with other traffic at the 'legacy' bottleneck.

Certain network operators might choose to restrict access to the L4S class, perhaps only to customers who have paid a premium. In the packet classifier (item 2 in Figure 1), they could identify such customers using some other field than ECN (e.g. source address range), and just ignore the L4S identifier for non-paying customers. This would ensure that the L4S identifier survives end-to-end even though the service does not have to be supported at every hop. Such arrangements would only require simple registered/not-registered packet classification, rather than the managed application-specific traffic policing against customer-specific traffic contracts that Diffserv requires.

8.2. 'Latency Friendliness'

The L4S service does rely on self-constraint - not in terms of limiting capacity usage, but in terms of limiting burstiness. It is believed that standardisation of dynamic behaviour (cf. TCP slow-start) and self-interest will be sufficient to prevent transports from sending excessive bursts of L4S traffic, given the application's own latency will suffer most from such behaviour.

Whether burst policing becomes necessary remains to be seen. Without it, there will be potential for attacks on the low latency of the L4S service. However it may only be necessary to apply such policing reactively, e.g. punitively targeted at any deployments of new bursty malware.

8.3. ECN Integrity

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 pseudorandomly 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 the receiver supports the nonce, it can prove that it is not suppressing feedback by reflecting its knowledge of the sequence back to the sender. The nonce was proposed on the assumption that receivers might be more likely to cheat congestion control than senders (although senders also have a motive to cheat).

If L4S uses the ECT(1) codepoint of ECN for packet classification, it will have to obsolete the experimental nonce. 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. For instance:

- o the sender can test the integrity of the receiver's feedback by occasionally setting the IP-ECN field to a value normally only set by the network. Then it can test whether the receiver's feedback faithfully reports what it expects [I-D.moncaster-tcpm-rcv-cheat]. This method consumes no extra codepoints. It works for loss and it will work for ECN feedback in any transport protocol suitable for L4S. However, it shares the same assumption as the nonce; that the sender is not cheating and it is motivated to prevent the receiver cheating;
- o 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. ConEx is only currently defined for IPv6 and consumes a destination option header. It has been implemented, but not deployed as far as is known.

9. Acknowledgements

10. References

10.1. Normative References

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

10.2. Informative References

[Alizadeh-stability]

Alizadeh, M., Javanmard, A., and B. Prabhakar, "Analysis of DCTCP: Stability, Convergence, and Fairness", ACM SIGMETRICS 2011 , June 2011.

[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)

[Hohlfeld14]

Hohlfeld , O., Pujol, E., Ciucu, F., Feldmann, A., and P. Barford, "A QoE Perspective on Sizing Network Buffers", Proc. ACM Internet Measurement Conf (IMC'14) hmm, November 2014.

[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-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-02 (work in progress), October 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.khademi-tcpm-alternativebackoff-ecn]
Khademi, N., Welzl, M., Armitage, G., and G. Fairhurst, "TCP Alternative Backoff with ECN (ABE)", draft-khademi-tcpm-alternativebackoff-ecn-01 (work in progress), October 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.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.
- [I-D.you-encrypted-traffic-management]
You, J. and C. Xiong, "The Effect of Encrypted Traffic on the QoS Mechanisms in Cellular Networks", draft-you-encrypted-traffic-management-00 (work in progress), October 2015.
- [Mathis09]
Mathis, M., "Relentless Congestion Control", PFLDNeT'09 , May 2009, <http://www.hpcc.jp/pfldnet2009/Program_files/1569198525.pdf>.
- [NewCC_Proc]
Eggert, L., "Experimental Specification of New Congestion Control Algorithms", IETF Operational Note ion-tsv-alt-cc, July 2007.

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

[TCP-sub-mss-w]

Briscoe, B. and K. De Schepper, "Scaling TCP's Congestion Window for Small Round Trip Times", BT Technical Report TR-TUB8-2015-002, May 2015, <<http://www.bobbriscoe.net/projects/latency/sub-mss-w.pdf>>.

[TCPPrague]

Briscoe, B., "Notes: DCTCP evolution 'bar BoF': Tue 21 Jul 2015, 17:40, Prague", tcpprague mailing list archive, July 2015.

Appendix A. Required features for scalable transport protocols to be safely deployable in the Internet (a.k.a. TCP Prague requirements)

This list contains a list of features, mechanisms and modifications from currently defined behaviour for scalable Transport protocols so that they can be safely deployed over the public Internet. This list of requirements was produced at an ad hoc meeting during IETF-94 in Prague [TCPPrague].

One of such scalable transport protocols is DCTCP, currently specified in [I-D.ietf-tcpm-dctcp]. In its current form, DCTCP is specified to be deployable in controlled environments and deploying it in the public Internet would lead to a number of issues, both from the safety and the performance perspective. In this section, we describe the modifications and additional mechanisms that are required for its deployment over the global Internet. We use DCTCP as a base, but it is likely that most of these requirements equally apply to other scalable transport protocols.

We next provide a brief description of each required feature.

Requirement #4.1: Fall back to Reno/Cubic congestion control on packet loss.

Description: In case of packet loss, the scalable transport MUST react as classic TCP (whatever the classic version of TCP is running in the host, e.g. Reno, Cubic).

Motivation: As part of the safety conditions for deploying a scalable transport over the public Internet is to make sure that it behaves

properly when some or all the network devices connecting the two endpoints that implement the scalable transport have not been upgraded. In particular, it may be the case that some of the switches along the path between the two endpoints may only react to congestion by dropping packets (i.e. no ECN marking). It is important that in these cases, the scalable transport react to the congestion signal in the form of a packet drop similarly to classic TCP.

In the particular case of DCTCP, the current DCTCP specification states that "It is RECOMMENDED that an implementation deal with loss episodes in the same way as conventional TCP." For safe deployment in the public Internet of a scalable transport, the above requirement needs to be defined as a MUST.

Packet loss, while rare, may also occur in the case that the bottleneck is L4S capable. In this case, the sender may receive a high number of packets marked with the CE bit set and also experience a loss. Current DCTCP implementations react differently to this situation. At least one implementation reacts only to the drop signal (e.g. by halving the CWND) and at least another DCTCP implementation reacts to both signals (e.g. by halving the CWND due to the drop and also further reducing the CWND based on the proportion of marked packet). We believe that further experimentation is needed to understand what is the best behaviour for the public Internet, which may or not be one of the existent implementations.

Requirement #4.2: Fall back to Reno/Cubic congestion control on classic ECN bottlenecks.

Description: The scalable transport protocol SHOULD/MAY? behave as classic TCP with classic ECN if the path contains a legacy bottleneck which marks both `ect(0)` and `ect(1)` in the same way as drop (non L4S, but ECN capable bottleneck).

Motivation: Similarly to Requirement #3.1, this requirement is a safety condition in case L4S-capable endpoints are communicating over a path that contains one or more non-L4S but ECN capable switches and one of them happens to be the bottleneck. In this case, the scalable transport will attempt to fill in the buffer of the bottleneck switch up to the marking threshold and produce a small sawtooth around that operation point. The result is that the switch will set its operation point with the buffer full and all other non-scalable transports will be starved (as they will react reducing their CWND more aggressively than the scalable transport).

Scalable transports then **MUST** be able to detect the presence of a classic ECN bottleneck and fall back to classic TCP/classic ECN behaviour in this case.

Discussion: It is not clear at this point if it is possible to design a mechanism that always detect the aforementioned cases. One possibility is to base the detection on an increase on top of a minimum RTT, but it is not yet clear which value should trigger this. Having a delay based fall back response on L4S may as well be beneficial for preserving low latency without legacy network nodes. Even if it possible to design such a mechanism, it may well be that it would encompass additional complexity that implementers may consider unnecessary. The need for this mechanism depends on the extent of classic ECN deployment.

Requirement #4.3: Reduce RTT dependence

Description: Scalable transport congestion control algorithms **MUST** reduce or eliminate the RTT bias within the range of RTTs available.

Motivation: Classic TCP's throughput is known to be inversely proportional to RTT. One would expect flows over very low RTT paths to nearly starve flows over larger RTTs. However, because Classic TCP induces a large queue, it has never allowed a very low RTT path to exist, so far. For instance, consider two paths with base RTT 1ms and 100ms. If Classic TCP induces a 20ms queue, it turns these RTTs into 21ms and 120ms leading to a throughput ratio of about 1:6. Whereas if a Scalable TCP induces only a 1ms queue, the ratio is 2:101. Therefore, with small queues, long RTT flows will essentially starve.

Scalable transport protocol **MUST** then accommodate flows across the range of RTTs enabled by the deployment of L4S service over the public Internet.

Requirement #4.4: Scaling down the congestion window.

Description: Scalable transports **MUST** be responsive to congestion when RTTs are significantly smaller than in the current public Internet.

Motivation: As currently specified, the minimum CWND of TCP (and the scalable extensions such as DCTCP), is set to 2 MSS. Once this minimum CWND is reached, the transport protocol ceases to react to congestion signals (the CWND is not further reduced beyond this minimum size).

L4S mechanisms reduce significantly the queueing delay, achieving smaller RTTs over the Internet. For the same CWND, smaller RTTs imply higher transmission rates. The result is that when scalable transport are used and small RTTs are achieved, the minimum value of the CWND currently defined in 2 MSS may still result in a high transmission rate for a large number of common scenarios. For example, as described in [TCP-sub-mss-w], consider a residential setting with an broadband Internet access of 40Mbps. Suppose now a number of equal TCP flows running in parallel with the Internet access link being the bottleneck. Suppose that for these flows, the RTT is 6ms and the MSS is 1500B. The minimum transmission rate supported by TCP in this scenario is when CWND is set to 2 MSS, which results in 4Mbps for each flow. This means that in this scenario, if the number of flows is higher than 10, the congestion control ceases to be responsive and starts to build up a queue in the network.

In order to address this issue, the congestion control mechanism for scalable transports MUST be responsive for the new range of RTT resulting from the decrease of the queueing delay.

There are several ways how this can be achieved. One possible sub-MSS window mechanism is described in [TCP-sub-mss-w].

In addition to the safety requirements described before, there are some optimizations that while not required for the safe deployment of scalable transports over the public Internet, would results in an optimized performance. We describe them next.

Optimization #5.1: Setting ECT in SYN, SYN/ACK and pure ACK packets.

Description: Scalable transport SHOULD set the ECT bit in SYN, SYN/ACK and pure ACK packets.

Motivation: Failing to set the ECT bit in SYN, SYN/ACK or ACK packets results in these packets being more likely dropped during congestion events. Dropping SYN and SYN/ACK packets is particularly bad for performance as the retransmission timers for these packets are large. [RFC3168] prevents from marking these packets due to security reasons. The arguments provided should be revisited in the the context of L4S and evaluate if avoiding marking these packets is still the best approach.

Optimization #5.2: Faster than additive increase.

Description: Scalable transport MAY support faster than additive increase in the congestion avoidance phase.

Motivation: As currently defined, DCTCP supports additive increase in congestion avoidance phase. It would be beneficial for performance to update the congestion control algorithm to increase the CWND more than 1 MSS per RTT during the congestion avoidance phase. In the context of L4S such mechanism, must also provide fairness with other classes of traffic, including classic TCP and possibly scalable TCP that uses additive increase.

Optimization #5.3: Faster convergence to fairness.

Description: Scalable transport SHOULD converge to a fair share allocation of the available capacity as fast as classic TCP or faster.

Motivation: The time required for a new flow to obtain its fair share of the capacity of the bottleneck when there are already ongoing flows using up all the bottleneck capacity is higher in the case of DCTCP than in the case of classic TCP (about a factor of 1,5 and 2 larger according to [Alizadeh-stability]). This is detrimental in general, but it is very harmful for short flows, which performance can be worse than the one obtained with classic TCP. For this reason it is desirable that scalable transport provide convergence times no larger than classic TCP.

Appendix B. Standardization items

The following table includes all the items that should be standardized to provide a full L4S architecture.

The table is too wide for the ASCII draft format, so it has been split into two, with a common column of row index numbers on the left.

The columns in the second part of the table have the following meanings:

WG: The IETF WG most relevant to this requirement. The "tcpm/iccr" combination refers to the procedure typically used for congestion control changes, where tcpm owns the approval decision, but uses the iccr for expert review [NewCC_Proc];

TCP: Applicable to all forms of TCP congestion control;

DCTCP: Applicable to Data Centre TCP as currently used (in controlled environments);

DCTCP bis: Applicable to an future Data Centre TCP congestion control intended for controlled environments;

XXX Prague: Applicable to a Scalable variant of XXX (TCP/SCTP/RMCAT) congestion control.

Req #	Requirement	Reference
0	ARCHITECTURE	
1	L4S IDENTIFIER	[I-D.briscoe-tsvwg-ecn-l4s-id]
2	DUAL QUEUE AQM	[I-D.briscoe-aqm-dualq-coupled]
3	Suitable ECN Feedback	[I-D.ietf-tcpm-accurate-ecn], [I-D.stewart-tsvwg-sctpecn].
	SCALABLE TRANSPORT - SAFETY ADDITIONS	
4-1	Fall back to Reno/Cubic on loss	[I-D.ietf-tcpm-dctcp]
4-2	Fall back to Reno/Cubic if classic ECN bottleneck detected	
4-3	Reduce RTT-dependence	
4-4	Scaling TCP's Congestion Window for Small Round Trip Times	[TCP-sub-mss-w]
	SCALABLE TRANSPORT - PERFORMANCE ENHANCEMENTS	
5-1	Setting ECT in SYN, SYN/ACK and pure ACK packets	draft-bagnulo-tsvwg-generalized-ECN
5-2	Faster-than-additive increase	
5-3	Less drastic exit from slow-start	

#	WG	TCP	DCTCP	DCTCP-bis	TCP Prague	SCTP Prague	RMCAT Prague
0	tsvwg?	Y	Y	Y	Y	Y	Y
1	tsvwg?			Y	Y	Y	Y
2	aqm?	n/a	n/a	n/a	n/a	n/a	n/a
3	tcpm	Y	Y	Y	Y	n/a	n/a
4-1	tcpm		Y	Y	Y	Y	Y
4-2	tcpm/ iccrgr?				Y	Y	?
4-3	tcpm/ iccrgr?			Y	Y	Y	?
4-4	tcpm	Y	Y	Y	Y	Y	?
5-1	tsvwg	Y	Y	Y	Y	n/a	n/a
5-2	tcpm/ iccrgr?			Y	Y	Y	?
5-3	tcpm/ iccrgr?			Y	Y	Y	?

Authors' Addresses

Bob Briscoe (editor)
Simula Research Lab

Email: ietf@bobbbriscoe.net
URI: <http://bobbbriscoe.net/>

Koen De Schepper
Nokia Bell Labs
Antwerp
Belgium

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

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
Spain

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: <http://www.it.uc3m.es>

Transport Area Working Group
Internet-Draft
Intended status: Informational
Expires: October 1, 2017

B. Briscoe, Ed.
Simula Research Lab
K. De Schepper
Nokia Bell Labs
M. Bagnulo Braun
Universidad Carlos III de Madrid
March 30, 2017

Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service:
Architecture
draft-briscoe-tsvwg-l4s-arch-02

Abstract

This document describes the L4S architecture for the provision of a new service that the Internet could provide to eventually replace best efforts for all traffic: Low Latency, Low Loss, Scalable throughput (L4S). It is becoming common for all (or most) applications being run by a user at any one time to require low latency. However, the only solution the IETF can offer for ultra-low queuing delay is Diffserv, which only favours a minority of packets at the expense of others. In extensive testing the new L4S service keeps average queuing delay under a millisecond for all applications even under very heavy load, without sacrificing utilization; and it keeps congestion loss to zero. It is becoming widely recognized that adding more access capacity gives diminishing returns, because latency is becoming the critical problem. Even with a high capacity broadband access, the reduced latency of L4S remarkably and consistently improves performance under load for applications such as interactive video, conversational video, voice, Web, gaming, instant messaging, remote desktop and cloud-based apps (even when all being used at once over the same access link). The insight is that the root cause of queuing delay is in TCP, not in the queue. By fixing the sending TCP (and other transports) queuing latency becomes so much better than today that operators will want to deploy the network part of L4S to enable new products and services. Further, the network part is simple to deploy - incrementally with zero-config. Both parts, sender and network, ensure coexistence with other legacy traffic. At the same time L4S solves the long-recognized problem with the future scalability of TCP throughput.

This document describes the L4S architecture, briefly describing the different components and how the work together to provide the aforementioned enhanced Internet service.

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 1, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. L4S architecture overview	4
3. Terminology	6
4. L4S architecture components	7
5. Rationale	9
5.1. Why These Primary Components?	9
5.2. Why Not Alternative Approaches?	10
6. Applicability	12
6.1. Use Cases	13
6.2. Deployment Considerations	14
6.2.1. Deployment Topology	15
6.2.2. Deployment Sequences	16
6.2.3. L4S Flow but Non-L4S Bottleneck	18
6.2.4. Other Potential Deployment Issues	19

7. IANA Considerations	19
8. Security Considerations	19
8.1. Traffic (Non-)Policing	19
8.2. 'Latency Friendliness'	20
8.3. Policing Prioritized L4S Bandwidth	20
8.4. ECN Integrity	21
9. Acknowledgements	22
10. References	22
10.1. Normative References	22
10.2. Informative References	22
Appendix A. Required features for scalable transport protocols to be safely deployable in the Internet (a.k.a. TCP Prague requirements)	26
Appendix B. Standardization items	30
Authors' Addresses	33

1. Introduction

It is increasingly common for all of a user's applications at any one time to require low delay: interactive Web, Web services, voice, conversational video, interactive video, instant messaging, online gaming, remote desktop and cloud-based applications. In the last decade or so, much has been done to reduce propagation delay by placing caches or servers closer to users. However, queuing remains a major, albeit intermittent, component of latency. When present it typically doubles the path delay from that due to the base speed-of-light. Low loss is also important because, for interactive applications, losses translate into even longer retransmission delays.

It has been demonstrated that, once access network bit rates reach levels now common in the developed world, increasing capacity offers diminishing returns if latency (delay) is not addressed. Differentiated services (Diffserv) offers Expedited Forwarding [RFC3246] for some packets at the expense of others, but this is not applicable when all (or most) of a user's applications require low latency.

Therefore, the goal is an Internet service with ultra-Low queueing Latency, ultra-Low Loss and Scalable throughput (L4S) - for all traffic. A service for all traffic will need none of the configuration or management baggage (traffic policing, traffic contracts) associated with favouring some packets over others. This document describes the L4S architecture for achieving that goal.

It must be said that queuing delay only degrades performance infrequently [Hohlfeld14]. It only occurs when a large enough capacity-seeking (e.g. TCP) flow is running alongside the user's

traffic in the bottleneck link, which is typically in the access network. Or when the low latency application is itself a large capacity-seeking flow (e.g. interactive video). At these times, the performance improvement must be so remarkable that network operators will be motivated to deploy it.

Active Queue Management (AQM) is part of the solution to queuing under load. AQM improves performance for all traffic, but there is a limit to how much queuing delay can be reduced by solely changing the network; without addressing the root of the problem.

The root of the problem is the presence of standard TCP congestion control (Reno [RFC5681]) or compatible variants (e.g. TCP Cubic [I-D.ietf-tcpm-cubic]). We shall call this family of congestion controls 'Classic' TCP. It has been demonstrated that if the sending host replaces Classic TCP with a 'Scalable' alternative, when a suitable AQM is deployed in the network the performance under load of all the above interactive applications can be stunningly improved. For instance, queuing delay under heavy load with the example DCTCP/DualQ solution cited below is roughly 1 millisecond (1 ms) at the 99th percentile without losing link utilization. This compares with 5 to 20 ms on average with a Classic TCP and current state-of-the-art AQMs such as fq_CoDel [I-D.ietf-aqm-fq-codel] or PIE [RFC8033]. Also, with a Classic TCP, 5 ms of queuing is usually only possible by losing some utilization.

It has been convincingly demonstrated [DCTtH15] that it is possible to deploy such an L4S service alongside the existing best efforts service so that all of a user's applications can shift to it when their stack is updated. Access networks are typically designed with one link as the bottleneck for each site (which might be a home, small enterprise or mobile device), so deployment at a single node should give nearly all the benefit. The L4S approach requires a number of mechanisms in different parts of the Internet to fulfill its goal. This document presents the L4S architecture, by describing the different components and how they interact to provide the scalable low-latency, low-loss, Internet service.

2. L4S architecture overview

There are three main components to the L4S architecture (illustrated in Figure 1):

- 1) Network: The L4S service traffic needs to be isolated from the queuing latency of the Classic service traffic. However, the two should be able to freely share a common pool of capacity. This is because there is no way to predict how many flows at any one time might use each service and capacity in access networks is too

scarce to partition into two. So a 'semi-permeable' membrane is needed that partitions latency but not bandwidth. The Dual Queue Coupled AQM [I-D.briscoe-aqm-dualq-coupled] is an example of such a semi-permeable membrane.

Per-flow queuing such as in [I-D.ietf-aqm-fq-codel] could be used, but it partitions both latency and bandwidth between every end-to-end flow. So it is rather overkill, which brings disadvantages (see Section 5.2), not least that thousands of queues are needed when two are sufficient.

- 2) Protocol: A host needs to distinguish L4S and Classic packets with an identifier so that the network can classify them into their separate treatments. [I-D.briscoe-tsvwg-ecn-l4s-id] considers various alternative identifiers, and concludes that all alternatives involve compromises, but the ECT(1) codepoint of the ECN field is a workable solution.
- 3) Host: Scalable congestion controls already exist. They solve the scaling problem with TCP first pointed out in [RFC3649]. The one used most widely (in controlled environments) is Data Centre TCP (DCTCP [I-D.ietf-tcpm-dctcp]), which has been implemented and deployed in Windows Server Editions (since 2012), in Linux and in FreeBSD. Although DCTCP as-is 'works' well over the public Internet, most implementations lack certain safety features that will be necessary once it is used outside controlled environments like data centres (see later). A similar scalable congestion control will also need to be transplanted into protocols other than TCP (SCTP, RTP/RTCP, RMCAT, etc.)

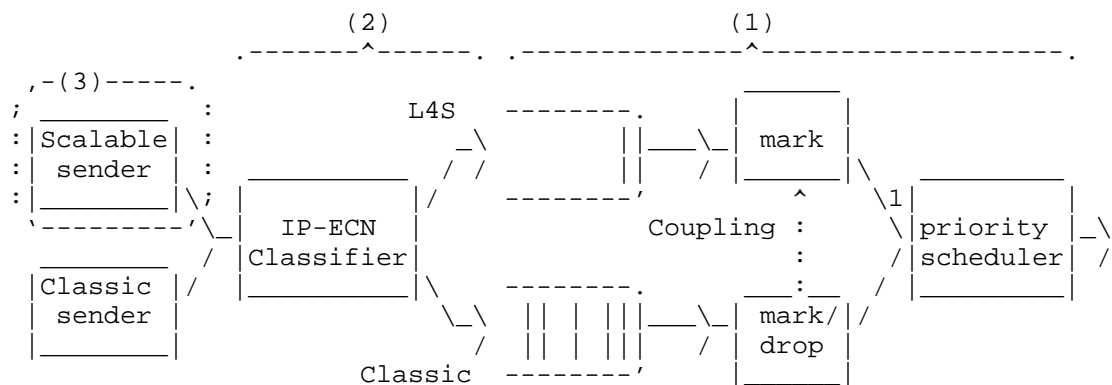


Figure 1: Components of an L4S Solution: 1) Isolation in separate network queues; 2) Packet Identification Protocol; and 3) Scalable Sending Host

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 [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. COMMENT: Since this will be an information document, This should be removed.

Classic service: The 'Classic' service is intended for all the congestion control 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).

Scalable Congestion Control: A congestion control where flow rate is inversely proportional to the level of congestion signals. Then, as flow rate scales, the number of congestion signals per round trip remains invariant, maintaining the same degree of control. For instance, DCTCP averages 2 congestion signals per round-trip whatever the flow rate.

Classic Congestion Control: A congestion control with a flow rate compatible with standard TCP Reno [RFC5681]. With Classic congestion controls, as capacity increases enabling higher flow rates, the number of round trips between congestion signals (losses or ECN marks) rises in proportion to the flow rate. So control of queuing and/or utilization becomes very slack. For instance, with 1500 B packets and an RTT of 18 ms, as TCP Reno flow rate increases from 2 to 100 Mb/s the number of round trips between congestion signals rises proportionately, from 2 to 100.

The default congestion control in Linux (TCP Cubic) is Reno-compatible for most scenarios expected for some years. For instance, with a typical domestic round-trip time (RTT) of 18ms, TCP Cubic only switches out of Reno-compatibility mode once the flow rate approaches 1 Gb/s. For a typical data centre RTT of 1 ms, the switch-over point is theoretically 1.3 Tb/s. However, with a less common transcontinental RTT of 100 ms, it only remains

Reno-compatible up to 13 Mb/s. All examples assume 1,500 B packets.

Classic ECN: The original proposed standard Explicit Congestion Notification (ECN) protocol [RFC3168], which requires ECN signals to be treated the same as drops, both when generated in the network and when responded to by the sender.

Site: A home, mobile device, small enterprise or campus, where the network bottleneck is typically the access link to the site. Not all network arrangements fit this model but it is a useful, widely applicable generalisation.

4. L4S architecture components

The L4S architecture is composed by the following elements.

Protocols: The L4S architecture encompass the two protocol changes that we describe next:

- a. [I-D.briscoe-tsvwg-ecn-l4s-id] recommends ECT(1) is used as the identifier to classify L4S and Classic packets into their separate treatments, as required by [RFC4774].
- b. An essential aspect of a scalable congestion control is the use of explicit congestion signals rather than losses, because the signals need to be sent immediately and frequently--too often to use drops. 'Classic' ECN [RFC3168] requires an ECN signal to be treated the same as a drop, both when it is generated in the network and when it is responded to by hosts. L4S allows networks and hosts to support two separate meanings for ECN. So the standards track [RFC3168] will need to be updated to allow ECT(1) packets to depart from the 'same as drop' constraint.

[I-D.ietf-tsvwg-ecn-experimentation] has been prepared as a standards track update to relax specific requirements in RFC 3168 (and certain other standards track RFCs), which clears the way for the above experimental changes proposed for L4S.

[I-D.ietf-tsvwg-ecn-experimentation] also obsoletes the original experimental assignment of the ECT(1) codepoint as an ECN nonce [RFC3540] (it was never deployed, and it offers no security benefit now that deployment is optional).

Network components: The Dual Queue Coupled AQM has been specified as generically as possible [I-D.briscoe-aqm-dualq-coupled] as a 'semi-permeable' membrane without specifying the particular AQMs to use in the two queues. An informational appendix of the draft is provided for pseudocode examples of different possible AQM approaches.

Initially a zero-config variant of RED called Curvy RED was implemented, tested and documented. The aim is for designers to be free to implement diverse ideas. So the brief normative body of the draft only specifies the minimum constraints an AQM needs to comply with to ensure that the L4S and Classic services will coexist. For instance, a variant of PIE called Dual PI Squared [PI2] has been implemented and found to perform better over a wide range of conditions, so it has been documented in a second appendix of [I-D.briscoe-aqm-dualq-coupled].

Host mechanisms: The L4S architecture includes a number of mechanisms in the end host that we enumerate next:

- a. Data Centre TCP is the most widely used example of a scalable congestion control. It is being documented in the TCPM WG as an informational record of the protocol currently in use [I-D.ietf-tcpm-dctcp]. It will be necessary to define a number of safety features for a variant usable on the public Internet. A draft list of these, known as the TCP Prague requirements, has been drawn up (see Appendix A). The list also includes some optional performance improvements.
- b. Transport protocols other than TCP use various congestion controls designed to be friendly with Classic TCP. Before they can use the L4S service, it will be necessary to implement scalable variants of each of these transport behaviours. The following standards track RFCs currently define these protocols: ECN in TCP [RFC3168], in SCTP [RFC4960], in RTP [RFC6679], and in DCCP [RFC4340]. Not all are in widespread use, but those that are will eventually need to be updated to allow a different congestion response, which they will have to indicate by using the ECT(1) codepoint. Scalable variants are under consideration for some new transport protocols that are themselves under development, e.g. QUIC [I-D.johansson-quic-ecn] and certain real-time media congestion avoidance techniques (RMCAT) protocols.
- c. ECN feedback is sufficient for L4S in some transport protocols (RTCP, DCCP) but not others:
 - * For the case of TCP, the feedback protocol for ECN embeds the assumption from Classic ECN that it is the same as drop, making it unusable for a scalable TCP. Therefore, the implementation of TCP receivers will have to be upgraded [RFC7560]. Work to standardize more accurate ECN feedback for TCP (AccECN [I-D.ietf-tcpm-accurate-ecn]) is already in progress.

- * ECN feedback is only roughly sketched in an appendix of the SCTP specification. A fuller specification has been proposed [I-D.stewart-tsvwg-sctpecn], which would need to be implemented and deployed before SCTCP could support L4S.

5. Rationale

5.1. Why These Primary Components?

Explicit congestion signalling (protocol): Explicit congestion signalling is a key part of the L4S approach. In contrast, use of drop as a congestion signal creates a tension because drop is both a useful signal (more would reduce delay) and an impairment (less would reduce delay). Explicit congestion signals can be used many times per round trip, to keep tight control, without any impairment. Under heavy load, even more explicit signals can be applied so the queue can be kept short whatever the load. Whereas state-of-the-art AQMs have to introduce very high packet drop at high load to keep the queue short. Further, TCP's sawtooth reduction can be smaller, and therefore return to the operating point more often, without worrying that this causes more signals (one at the top of each smaller sawtooth). The consequent smaller amplitude sawteeth fit between a very shallow marking threshold and an empty queue, so delay variation can be very low, without risk of under-utilization.

All the above makes it clear that explicit congestion signalling is only advantageous for latency if it does not have to be considered 'the same as' drop (as required with Classic ECN [RFC3168]). Therefore, in a DualQ AQM, the L4S queue uses a new L4S variant of ECN that is not equivalent to drop [I-D.briscoe-tsvwg-ecn-l4s-id], while the Classic queue uses either classic ECN [RFC3168] or drop, which are equivalent.

Before Classic ECN was standardized, there were various proposals to give an ECN mark a different meaning from drop. However, there was no particular reason to agree on any one of the alternative meanings, so 'the same as drop' was the only compromise that could be reached. RFC 3168 contains a statement that:

"An environment where all end nodes were ECN-Capable could allow new criteria to be developed for setting the CE codepoint, and new congestion control mechanisms for end-node reaction to CE packets. However, this is a research issue, and as such is not addressed in this document."

Latency isolation with coupled congestion notification (network):

Using just two queues is not essential to L4S (more would be possible), but it is the simplest way to isolate all the L4S traffic that keeps latency low from all the legacy Classic traffic that does not.

Similarly, coupling the congestion notification between the queues is not necessarily essential, but it is a clever and simple way to allow senders to determine their rate, packet-by-packet, rather than be overridden by a network scheduler. Because otherwise a network scheduler would have to inspect at least transport layer headers, and it would have to continually assign a rate to each flow without any easy way to understand application intent.

L4S packet identifier (protocol): Once there are at least two separate treatments in the network, hosts need an identifier at the IP layer to distinguish which treatment they intend to use.

Scalable congestion notification (host): A scalable congestion control keeps the signalling frequency high so that rate variations can be small when signalling is stable, and rate can track variations in available capacity as rapidly as possible otherwise.

5.2. Why Not Alternative Approaches?

All the following approaches address some part of the same problem space as L4S. In each case, it is shown that L4S complements them or improves on them, rather than being a mutually exclusive alternative:

Diffserv: Diffserv addresses the problem of bandwidth apportionment for important traffic as well as queuing latency for delay-sensitive traffic. L4S solely addresses the problem of queuing latency (as well as loss and throughput scaling). Diffserv will still be necessary where important traffic requires priority (e.g. for commercial reasons, or for protection of critical infrastructure traffic). Nonetheless, if there are Diffserv classes for important traffic, the L4S approach can provide low latency for all traffic within each Diffserv class (including the case where there is only one Diffserv class).

Also, as already explained, Diffserv only works for a small subset of the traffic on a link. It is not applicable when all the applications in use at one time at a single site (home, small business or mobile device) require low latency. Also, because L4S is for all traffic, it needs none of the management baggage (traffic policing, traffic contracts) associated with favouring some packets over others. This baggage has held Diffserv back from widespread end-to-end deployment.

State-of-the-art AQMs: AQMs such as PIE and fq_CoDel give a significant reduction in queuing delay relative to no AQM at all. The L4S work is intended to complement these AQMs, and we definitely do not want to distract from the need to deploy them as widely as possible. Nonetheless, without addressing the large saw-tooth rate variations of Classic congestion controls, AQMs alone cannot reduce queuing delay too far without significantly reducing link utilization. The L4S approach resolves this tension by ensuring hosts can minimize the size of their sawteeth without appearing so aggressive to legacy flows that they starve.

Per-flow queuing: Similarly per-flow queuing is not incompatible with the L4S approach. However, one queue for every flow can be thought of as overkill compared to the minimum of two queues for all traffic needed for the L4S approach. The overkill of per-flow queuing has side-effects:

- A. fq makes high performance networking equipment costly (processing and memory) - in contrast dual queue code can be very simple;
- B. fq requires packet inspection into the end-to-end transport layer, which doesn't sit well alongside encryption for privacy - in contrast a dual queue only operates at the IP layer;
- C. fq isolates the queuing of each flow from the others and it prevents any one flow from consuming more than $1/N$ of the capacity. In contrast, all L4S flows are expected to keep the queue shallow, and policing of individual flows to enforce this may be applied separately, as a policy choice.

An fq scheduler has to decide packet-by-packet which flow to schedule without knowing application intent. Whereas a separate policing function can be configured less strictly, so that senders can still control the instantaneous rate of each flow dependent on the needs of each application (e.g. variable rate video), giving more wriggle-room before a flow is deemed non-compliant. Also policing of queuing and of flow-rates can be applied independently.

Alternative Back-off ECN (ABE): Yet again, L4S is not an alternative to ABE but a complement that introduces much lower queuing delay. ABE [I-D.khademi-tcpm-alternativebackoff-ecn] alters the host behaviour in response to ECN marking to utilize a link better and give ECN flows a faster throughput, but it assumes the network still treats ECN and drop the same. Therefore ABE exploits any lower queuing delay that AQMs can provide. But as explained

above, AQMs still cannot reduce queuing delay too far without losing link utilization (for other non-ABE flows).

6. Applicability

A transport layer that solves the current latency issues will provide new service, product and application opportunities.

With the L4S approach, the following existing applications will immediately experience significantly better quality of experience under load in the best effort class:

- o Gaming
- o VoIP
- o Video conferencing
- o Web browsing
- o (Adaptive) video streaming
- o Instant messaging

The significantly lower queuing latency also enables some interactive application functions to be offloaded to the cloud that would hardly even be usable today:

- o Cloud based interactive video
- o Cloud based virtual and augmented reality

The above two applications have been successfully demonstrated with L4S, both running together over a 40 Mb/s broadband access link loaded up with the numerous other latency sensitive applications in the previous list as well as numerous downloads. A panoramic video of a football stadium can be swiped and pinched so that on the fly a proxy in the cloud generates a sub-window of the match video under the finger-gesture control of each user. At the same time, a virtual reality headset fed from a 360 degree camera in a racing car has been demonstrated, where the user's head movements control the scene generated in the cloud. In both cases, with 7 ms end-to-end base delay, the additional queuing delay of roughly 1 ms is so low that it seems the video is generated locally. See <https://riteproject.eu/dctth/> for videos of these demonstrations.

Using a swiping finger gesture or head movement to pan a video are extremely demanding applications--far more demanding than VoIP.

Because human vision can detect extremely low delays of the order of single milliseconds when delay is translated into a visual lag between a video and a reference point (the finger or the orientation of the head).

If low network delay is not available, all fine interaction has to be done locally and therefore much more redundant data has to be downloaded. When all interactive processing can be done in the cloud, only the data to be rendered for the end user needs to be sent. Whereas, once applications can rely on minimal queues in the network, they can focus on reducing their own latency by only minimizing the application send queue.

6.1. Use Cases

The following use-cases for L4S are being considered by various interested parties:

- o Where the bottleneck is one of various types of access network: DSL, cable, mobile, satellite
 - * Radio links (cellular, WiFi) that are distant from the source are particularly challenging. The radio link capacity can vary rapidly by orders of magnitude, so it is often desirable to hold a buffer to utilise sudden increases of capacity;
 - * cellular networks are further complicated by a perceived need to buffer in order to make hand-overs imperceptible;
 - * Satellite networks generally have a very large base RTT, so even with minimal queuing, overall delay can never be extremely low;
 - * Nonetheless, it is certainly desirable not to hold a buffer purely because of the sawteeth of Classic TCP, when it is more than is needed for all the above reasons.
- o Private networks of heterogeneous data centres, where there is no single administrator that can arrange for all the simultaneous changes to senders, receivers and network needed to deploy DCTCP:
 - * a set of private data centres interconnected over a wide area with separate administrations, but within the same company
 - * a set of data centres operated by separate companies interconnected by a community of interest network (e.g. for the finance sector)

- * multi-tenant (cloud) data centres where tenants choose their operating system stack (Infrastructure as a Service - IaaS)
- o Different types of transport (or application) congestion control:
 - * elastic (TCP/SCTP);
 - * real-time (RTP, RMCAT);
 - * query (DNS/LDAP).
- o Where low delay quality of service is required, but without inspecting or intervening above the IP layer [I-D.you-encrypted-traffic-management]:
 - * mobile and other networks have tended to inspect higher layers in order to guess application QoS requirements. However, with growing demand for support of privacy and encryption, L4S offers an alternative. There is no need to select which traffic to favour for queuing, when L4S gives favourable queuing to all traffic.
- o If queuing delay is minimized, applications with a fixed delay budget can communicate over longer distances, or via a longer chain of service functions [RFC7665] or onion routers.

6.2. Deployment Considerations

The DualQ is, in itself, an incremental deployment framework for L4S AQMs so that L4S traffic can coexist with existing Classic "TCP-friendly" traffic. Section 6.2.1 explains why only deploying AQM in one node at each end of the access link will realize nearly all the benefit.

L4S involves both end systems and the network, so Section 6.2.2 suggests some typical sequences to deploy each part, and why there will be an immediate and significant benefit after deploying just one part.

If an ECN-enabled DualQ AQM has not been deployed at a bottleneck, an L4S flow is required to include a fall-back strategy to Classic behaviour. Section 6.2.3 describes how an L4S flow detects this, and how to minimize the effect of false negative detection.

6.2.1. Deployment Topology

Nonetheless, DualQ AQMs will not have to be deployed throughout the Internet before L4S will work for anyone. Operators of public Internet access networks typically design their networks so that the bottleneck will nearly always occur at one known (logical) link. This confines the cost of queue management technology to one place.

The case of mesh networks is different and will be discussed later. But the known bottleneck case is generally true for Internet access to all sorts of different 'sites', where the word 'site' includes home networks, small-to-medium sized campus or enterprise networks and even cellular devices (Figure 2). Also, this known-bottleneck case tends to be true whatever the access link technology; whether xDSL, cable, cellular, line-of-sight wireless or satellite.

Therefore, the full benefit of the L4S service should be available in the downstream direction when the DualQ AQM is deployed at the ingress to this bottleneck link (or links for multihomed sites). And similarly, the full upstream service will be available once the DualQ is deployed at the upstream ingress.

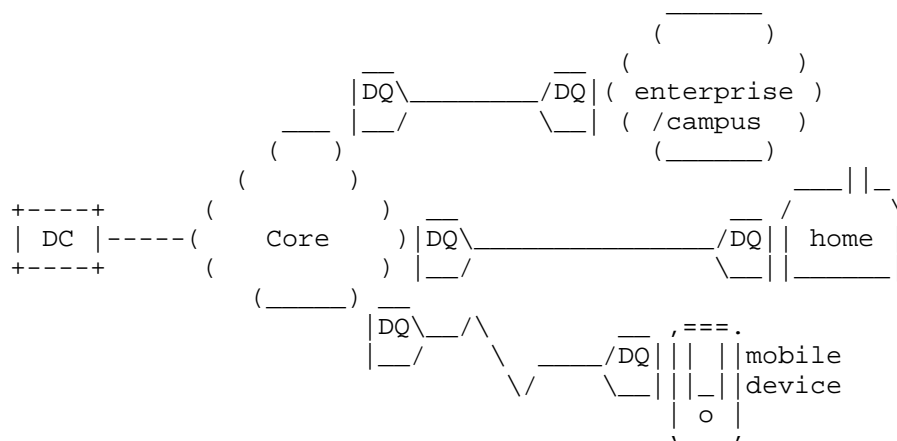


Figure 2: Likely location of DualQ Deployments in common access topologies

Deployment in mesh topologies depends on how over-booked the core is. If the core is non-blocking, or at least generously provisioned so that the edges are nearly always the bottlenecks, it would only be necessary to deploy the DualQ AQM at the edge bottlenecks. For example, some datacentre networks are designed with the bottleneck in

the hypervisor or host NICs, while others bottleneck at the top-of-rack switch (both the output ports facing hosts and those facing the core).

The DualQ would eventually also need to be deployed at any other persistent bottlenecks such as network interconnections, e.g. some public Internet exchange points and the ingress and egress to WAN links interconnecting datacentres.

6.2.2. Deployment Sequences

For any one L4S flow to work, it requires 3 parts to have been deployed. This was the same deployment problem that ECN faced [I-D.iab-protocol-transitions] so we have learned from this.

Firstly, L4S deployment exploits the fact that DCTCP already exists on many Internet hosts (Windows, FreeBSD and Linux); both servers and clients. Therefore, just deploying DualQ AQM at a network bottleneck immediately gives a working deployment of all the L4S parts. DCTCP needs some safety concerns to be fixed for general use over the public Internet (see Appendix A), but DCTCP is not on by default, so these issues can be managed within controlled deployments or controlled trials.

Secondly, the performance improvement with L4S is so significant that it enables new interactive services and products that were not previously possible. It is much easier for companies to initiate new work on deployment if there is budget for a new product trial. If, in contrast, there were only an incremental performance improvement (as with Classic ECN), spending on deployment tends to be much harder to justify.

Thirdly, the L4S identifier is defined so that initially network operators can enable L4S exclusively for certain customers or certain applications. But this is carefully defined so that it does not compromise future evolution towards L4S as an Internet-wide service. This is because the L4S identifier is defined not only as the end-to-end ECN field, but it can also optionally be combined with any other packet header or some status of a customer or their access link. Operators could do this anyway, even if it were not blessed by the IETF. However, it is best for the IETF to specify that they must use their own local identifier in combination with the IETF's identifier. Then, if an operator enables the optional local-use approach, they only have to remove this extra rule to make the service work Internet-wide - it will already traverse middleboxes, peerings, etc.

	Servers or proxies	Access link	Clients
1	DCTCP (existing)	DualQ AQM downstream	DCTCP (existing)
	WORKS DOWNSTREAM FOR CONTROLLED DEPLOYMENTS/TRIALS		
2	TCP Prague		AccECN (already in progress:DCTCP/BBR)
	FULLY	WORKS	DOWNSTREAM
3		DualQ AQM upstream	TCP Prague
	FULLY WORKS UPSTREAM AND DOWNSTREAM		

Figure 3: Example L4S Deployment Sequences

Figure 3 illustrates some example sequences in which the parts of L4S might be deployed. It consists of the following stages:

1. Here, the immediate benefit of a single AQM deployment can be seen, but limited to a controlled trial or controlled deployment. In this example downstream deployment is first, but in other scenarios the upstream might be go first. The DualQ AQM also greatly improves the downstream Classic service, assuming no other AQM has already been deployed.
2. In this stage, the name 'TCP Prague' is used to represent a variant of DCTCP that is safe to use in a production environment. If the application is primarily unidirectional, 'TCP Prague' is only needed at one end. Accurate ECN feedback (AccECN) [I-D.ietf-tcpm-accurate-ecn] is needed at the other end, but it is a generic ECN feedback facility that is already planned to be deployed for other purposes, e.g. DCTCP, BBR [BBR]. The two ends can be deployed in either order, because TCP Prague only enables itself if it has negotiated the use of AccECN feedback with the other end during the connection handshake. Thus, deployment on both ends (and in some cases only one) enables L4S trials to move to a production service, in one direction. This stage might be further motivated by performance improvements between DCTCP and TCP Prague Appendix A.
3. This is a two-move stage to enable L4S upstream. The DualQ or TCP Prague can be deployed in either order as already explained. To motivate the first of two independent moves, the deferred benefit of enabling new services after the second move has to be

worth it to cover the first mover's investment risk. As explained already, the potential for new services provides this motivation. The DualQ AQM also greatly improves the upstream Classic service, assuming no other AQM has already been deployed.

Note that other deployment sequences might occur. For instance: the upstream might be deployed first; a non-TCP protocol might be used end-to-end, e.g. QUIC, RMCAT; a body such as the 3GPP might require L4S to be implemented in 5G user equipment, or other random acts of kindness.

6.2.3. L4S Flow but Non-L4S Bottleneck

If L4S is enabled between two hosts but there is no L4S AQM at the bottleneck, any drop from the bottleneck will trigger the L4S sender to fall back to a 'TCP-Friendly' behaviour (Requirement #4.1 in Appendix A).

Unfortunately, as well as protecting legacy traffic, this rule degrades the L4S service whenever there is a loss, even if the loss was not from a non-DualQ bottleneck (false negative). And unfortunately, prevalent drop can be due to other causes, e.g.:

- o congestion loss at other transient bottlenecks, e.g. due to bursts in shallower queues;
- o transmission errors, e.g. due to electrical interference;
- o rate policing.

Three complementary approaches are in progress, but they are all currently research:

- o In TCP Prague, use a similar approach to BBR [BBR] to ignore selected losses. This could mask any of the above types of loss (requires consensus on how to safely interoperate with drop-based congestion controls).
- o A combination of RACK, reconfigured link retransmission and L4S could address transmission errors (no reference yet);
- o Hybrid ECN/drop policers (see Section 8.3).

L4S deployment scenarios that minimize these issues (e.g. over wireline networks) can proceed in parallel to this research, in the expectation that research success will continually widen L4S applicability.

In recent studies there has been no evidence of Classic ECN support in AQMs on the Internet. If Classic ECN support does materialize, a way to satisfy Requirement #4.2 in Appendix A will have to be added to TCP Prague.

6.2.4. Other Potential Deployment Issues

An L4S AQM uses the ECN field to signal congestion. So, in common with Classic ECN, if the AQM is within a tunnel or at a lower layer, correct functioning of ECN signalling requires correct propagation of the ECN field up the layers [I-D.ietf-tsvwg-ecn-encap-guidelines].

7. IANA Considerations

This specification contains no IANA considerations.

8. Security Considerations

8.1. Traffic (Non-)Policing

Because the L4S service can serve all traffic that is using the capacity of a link, it should not be necessary to police access to the L4S service. In contrast, Diffserv only works if some packets get less favourable treatment than others. So it has to use traffic policers to limit how much traffic can be favoured. In turn, traffic policers require traffic contracts between users and networks as well as pairwise between networks. Because L4S will lack all this management complexity, it is more likely to work end-to-end.

During early deployment (and perhaps always), some networks will not offer the L4S service. These networks do not need to police or remark L4S traffic - they just forward it unchanged as best efforts traffic, as they would already forward traffic with ECT(1) today. At a bottleneck, such networks will introduce some queuing and dropping. When a scalable congestion control detects a drop it will have to respond as if it is a Classic congestion control (see item 3-1 in Appendix A). This will ensure safe interworking with other traffic at the 'legacy' bottleneck, but it will degrade the L4S service to no better (but never worse) than classic best efforts, whenever a legacy (non-L4S) bottleneck is encountered on a path.

Certain network operators might choose to restrict access to the L4S class, perhaps only to customers who have paid a premium. Their packet classifier (item 2 in Figure 1) could identify such customers against some other field (e.g. source address range) as well as ECN. If only the ECN L4S identifier matched, but not the source address (say), the classifier could direct these packets (from non-paying customers) into the Classic queue. Allowing operators to use an

additional local classifier is intended to remove any incentive to bleach the L4S identifier. Then at least the L4S ECN identifier will be more likely to survive end-to-end even though the service may not be supported at every hop. Such arrangements would only require simple registered/not-registered packet classification, rather than the managed application-specific traffic policing against customer-specific traffic contracts that Diffserv requires.

8.2. 'Latency Friendliness'

The L4S service does rely on self-constraint - not in terms of limiting capacity usage, but in terms of limiting burstiness. It is hoped that standardisation of dynamic behaviour (cf. TCP slow-start) and self-interest will be sufficient to prevent transports from sending excessive bursts of L4S traffic, given the application's own latency will suffer most from such behaviour.

Whether burst policing becomes necessary remains to be seen. Without it, there will be potential for attacks on the low latency of the L4S service. However it may only be necessary to apply such policing reactively, e.g. punitively targeted at any deployments of new bursty malware.

8.3. Policing Prioritized L4S Bandwidth

As mentioned in Section 5.2, L4S should remove the need for low latency Diffserv classes. However, those Diffserv classes that give certain applications or users priority over capacity, would still be applicable. Then, within such Diffserv classes, L4S would often be applicable to give traffic low latency and low loss. Within such a class, the bandwidth available to a user or application is often limited by a rate policer. Similarly, in the default Diffserv class, rate policers are used to partition shared capacity.

A classic rate policer drops any packets exceeding a set rate, usually also giving a burst allowance (variant exist where the policer re-marks non-compliant traffic to a discard-eligible Diffserv codepoint, so they may be dropped elsewhere during contention). In networks that deploy L4S and use rate policers, it will be preferable to deploy a policer designed to be more friendly to the L4S service,

This is currently a research area. it might be achieved by setting a threshold where ECN marking is introduced, such that it is just under the policed rate or just under the burst allowance where drop is introduced. This could be applied to various types of policer, e.g. [RFC2697], [RFC2698] or the local (non-ConEx) variant of the ConEx congestion policer [I-D.briscoe-conex-policing]. Otherwise, whenever L4S traffic encounters a rate policer, it will experience drops and

the source will fall back to a Classic congestion control, thus losing all the benefits of L4S.

Further discussion of the applicability of L4S to the various Diffserv classes, and the design of suitable L4S rate policers.

8.4. ECN Integrity

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 pseudorandomly 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 the receiver supports the nonce, it can prove that it is not suppressing feedback by reflecting its knowledge of the sequence back to the sender. The nonce was proposed on the assumption that receivers might be more likely to cheat congestion control than senders (although senders also have a motive to cheat).

If L4S uses the ECT(1) codepoint of ECN for packet classification, it will have to obsolete the experimental nonce. 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. For instance:

- o the sender can test the integrity of the receiver's feedback by occasionally setting the IP-ECN field to a value normally only set by the network. Then it can test whether the receiver's feedback faithfully reports what it expects [I-D.moncaster-tcpm-rcv-cheat]. This method consumes no extra codepoints. It works for loss and it will work for ECN feedback in any transport protocol suitable for L4S. However, it shares the same assumption as the nonce; that the sender is not cheating and it is motivated to prevent the receiver cheating;
- o 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. ConEx is only currently defined for IPv6 and consumes a destination option header. It has been implemented, but not deployed as far as is known.

9. Acknowledgements

Thanks to Wes Eddy, Karen Nielsen and David Black for their useful review comments.

10. References

10.1. Normative References

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

10.2. Informative References

- [Alizadeh-stability] Alizadeh, M., Javanmard, A., and B. Prabhakar, "Analysis of DCTCP: Stability, Convergence, and Fairness", ACM SIGMETRICS 2011, June 2011.
- [BBR] Cardwell, N., Cheng, Y., Gunn, C., Yeganeh, S., and V. Jacobson, "BBR: Congestion-Based Congestion Control; Measuring bottleneck bandwidth and round-trip propagation time", ACM Queue (14)5, December 2016.
- [DCTth15] De Schepper, K., Bondarenko, O., Tsang, I., and B. Briscoe, "'Data Centre to the Home': Ultra-Low Latency for All", 2015, <http://www.bobbriscoe.net/projects/latency/dctth_preprint.pdf>.
- (Under submission)
- [Hohlfeld14] Hohlfeld, O., Pujol, E., Ciucu, F., Feldmann, A., and P. Barford, "A QoE Perspective on Sizing Network Buffers", Proc. ACM Internet Measurement Conf (IMC'14) hmmm, November 2014.
- [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-conex-policing]
Briscoe, B., "Network Performance Isolation using Congestion Policing", draft-briscoe-conex-policing-01 (work in progress), February 2014.
- [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-02 (work in progress), October 2016.
- [I-D.iab-protocol-transitions]
Thaler, D., "Planning for Protocol Adoption and Subsequent Transitions", draft-iab-protocol-transitions-08 (work in progress), March 2017.
- [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-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-04 (work in progress), February 2017.
- [I-D.ietf-tcpm-dctcp]
Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L., and G. Judd, "Datacenter TCP (DCTCP): TCP Congestion Control for Datacenters", draft-ietf-tcpm-dctcp-05 (work in progress), March 2017.
- [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-08 (work in progress), March 2017.

- [I-D.ietf-tsvwg-ecn-experimentation]
Black, D., "Explicit Congestion Notification (ECN) Experimentation", draft-ietf-tsvwg-ecn-experimentation-01 (work in progress), March 2017.
- [I-D.johansson-quic-ecn]
Johansson, I., "ECN support in QUIC", draft-johansson-quic-ecn-01 (work in progress), February 2017.
- [I-D.khademi-tcpm-alternativebackoff-ecn]
Khademi, N., Welzl, M., Armitage, G., and G. Fairhurst, "TCP Alternative Backoff with ECN (ABE)", draft-khademi-tcpm-alternativebackoff-ecn-01 (work in progress), October 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.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.
- [I-D.you-encrypted-traffic-management]
You, J. and C. Xiong, "The Effect of Encrypted Traffic on the QoS Mechanisms in Cellular Networks", draft-you-encrypted-traffic-management-00 (work in progress), October 2015.
- [Mathis09]
Mathis, M., "Relentless Congestion Control", PFLDNet'09 , May 2009, <http://www.hpcc.jp/pfldnet2009/Program_files/1569198525.pdf>.
- [NewCC_Proc]
Eggert, L., "Experimental Specification of New Congestion Control Algorithms", IETF Operational Note ion-tsv-alt-cc, July 2007.
- [PI2]
De Schepper, K., Bondarenko, O., Tsang, I., and B. Briscoe, "PI² : A Linearized AQM for both Classic and Scalable TCP", Proc. ACM CoNEXT 2016 pp.105-119, December 2016, <<http://dl.acm.org/citation.cfm?doid=2999572.2999578>>.

- [RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, DOI 10.17487/RFC2697, September 1999, <<http://www.rfc-editor.org/info/rfc2697>>.
- [RFC2698] Heinanen, J. and R. Guerin, "A Two Rate Three Color Marker", RFC 2698, DOI 10.17487/RFC2698, September 1999, <<http://www.rfc-editor.org/info/rfc2698>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<http://www.rfc-editor.org/info/rfc4960>>.
- [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>>.
- [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>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<http://www.rfc-editor.org/info/rfc7665>>.
- [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>>.
- [RFC8033] Pan, R., Natarajan, P., Baker, F., and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem", RFC 8033, DOI 10.17487/RFC8033, February 2017, <<http://www.rfc-editor.org/info/rfc8033>>.
- [TCP-sub-mss-w]
Briscoe, B. and K. De Schepper, "Scaling TCP's Congestion Window for Small Round Trip Times", BT Technical Report TR-TUB8-2015-002, May 2015, <<http://www.bobbriscoe.net/projects/latency/sub-mss-w.pdf>>.
- [TCPPrague]
Briscoe, B., "Notes: DCTCP evolution 'bar BoF': Tue 21 Jul 2015, 17:40, Prague", tcpprague mailing list archive , July 2015.

Appendix A. Required features for scalable transport protocols to be safely deployable in the Internet (a.k.a. TCP Prague requirements)

This list contains a list of features, mechanisms and modifications from currently defined behaviour for scalable Transport protocols so that they can be safely deployed over the public Internet. This list of requirements was produced at an ad hoc meeting during IETF-94 in Prague [TCPPrague].

One of such scalable transport protocols is DCTCP, currently specified in [I-D.ietf-tcpm-dctcp]. In its current form, DCTCP is specified to be deployable in controlled environments and deploying it in the public Internet would lead to a number of issues, both from the safety and the performance perspective. In this section, we describe the modifications and additional mechanisms that are required for its deployment over the global Internet. We use DCTCP as a base, but it is likely that most of these requirements equally apply to other scalable transport protocols.

We next provide a brief description of each required feature.

Requirement #4.1: Fall back to Reno/Cubic congestion control on packet loss.

Description: In case of packet loss, the scalable transport MUST react as classic TCP (whatever the classic version of TCP is running in the host, e.g. Reno, Cubic).

Motivation: As part of the safety conditions for deploying a scalable transport over the public Internet is to make sure that it behaves properly when some or all the network devices connecting the two endpoints that implement the scalable transport have not been upgraded. In particular, it may be the case that some of the switches along the path between the two endpoints may only react to congestion by dropping packets (i.e. no ECN marking). It is important that in these cases, the scalable transport react to the congestion signal in the form of a packet drop similarly to classic TCP.

In the particular case of DCTCP, the current DCTCP specification states that "It is RECOMMENDED that an implementation deal with loss episodes in the same way as conventional TCP." For safe deployment in the public Internet of a scalable transport, the above requirement needs to be defined as a MUST.

Packet loss, while rare, may also occur in the case that the bottleneck is L4S capable. In this case, the sender may receive a high number of packets marked with the CE bit set and also experience a loss. Current DCTCP implementations react differently to this situation. At least one implementation reacts only to the drop signal (e.g. by halving the CWND) and at least another DCTCP implementation reacts to both signals (e.g. by halving the CWND due to the drop and also further reducing the CWND based on the proportion of marked packet). We believe that further experimentation is needed to understand what is the best behaviour for the public Internet, which may or not be one of the existent implementations.

Requirement #4.2: Fall back to Reno/Cubic congestion control on classic ECN bottlenecks.

Description: The scalable transport protocol SHOULD/MAY? behave as classic TCP with classic ECN if the path contains a legacy bottleneck which marks both `ect(0)` and `ect(1)` in the same way as drop (non L4S, but ECN capable bottleneck).

Motivation: Similarly to Requirement #3.1, this requirement is a safety condition in case L4S-capable endpoints are communicating over a path that contains one or more non-L4S but ECN capable switches and one of them happens to be the bottleneck. In this case, the scalable transport will attempt to fill in the buffer of the bottleneck switch up to the marking threshold and produce a small sawtooth around that operation point. The result is that the switch will set its operation point with the buffer full and all other non-scalable transports will be starved (as they will react reducing their CWND more aggressively than the scalable transport).

Scalable transports then MUST be able to detect the presence of a classic ECN bottleneck and fall back to classic TCP/classic ECN behaviour in this case.

Discussion: It is not clear at this point if it is possible to design a mechanism that always detect the aforementioned cases. One possibility is to base the detection on an increase on top of a minimum RTT, but it is not yet clear which value should trigger this. Having a delay based fall back response on L4S may as well be beneficial for preserving low latency without legacy network nodes. Even if it possible to design such a mechanism, it may well be that it would encompass additional complexity that implementers may consider unnecessary. The need for this mechanism depends on the extent of classic ECN deployment.

Requirement #4.3: Reduce RTT dependence

Description: Scalable transport congestion control algorithms MUST reduce or eliminate the RTT bias within the range of RTTs available.

Motivation: Classic TCP's throughput is known to be inversely proportional to RTT. One would expect flows over very low RTT paths to nearly starve flows over larger RTTs. However, because Classic TCP induces a large queue, it has never allowed a very low RTT path to exist, so far. For instance, consider two paths with base RTT 1ms and 100ms. If Classic TCP induces a 20ms queue, it turns these RTTs into 21ms and 120ms leading to a throughput ratio of about 1:6. Whereas if a Scalable TCP induces only a 1ms queue, the ratio is

2:101. Therefore, with small queues, long RTT flows will essentially starve.

Scalable transport protocol MUST then accommodate flows across the range of RTTs enabled by the deployment of L4S service over the public Internet.

Requirement #4.4: Scaling down the congestion window.

Description: Scalable transports MUST be responsive to congestion when RTTs are significantly smaller than in the current public Internet.

Motivation: As currently specified, the minimum CWND of TCP (and the scalable extensions such as DCTCP), is set to 2 MSS. Once this minimum CWND is reached, the transport protocol ceases to react to congestion signals (the CWND is not further reduced beyond this minimum size).

L4S mechanisms reduce significantly the queueing delay, achieving smaller RTTs over the Internet. For the same CWND, smaller RTTs imply higher transmission rates. The result is that when scalable transport are used and small RTTs are achieved, the minimum value of the CWND currently defined in 2 MSS may still result in a high transmission rate for a large number of common scenarios. For example, as described in [TCP-sub-mss-w], consider a residential setting with an broadband Internet access of 40Mbps. Suppose now a number of equal TCP flows running in parallel with the Internet access link being the bottleneck. Suppose that for these flows, the RTT is 6ms and the MSS is 1500B. The minimum transmission rate supported by TCP in this scenario is when CWND is set to 2 MSS, which results in 4Mbps for each flow. This means that in this scenario, if the number of flows is higher than 10, the congestion control ceases to be responsive and starts to build up a queue in the network.

In order to address this issue, the congestion control mechanism for scalable transports MUST be responsive for the new range of RTT resulting from the decrease of the queueing delay.

There are several ways how this can be achieved. One possible sub-MSS window mechanism is described in [TCP-sub-mss-w].

In addition to the safety requirements described before, there are some optimizations that while not required for the safe deployment of scalable transports over the public Internet, would results in an optimized performance. We describe them next.

Optimization #5.1: Setting ECT in SYN, SYN/ACK and pure ACK packets.

Description: Scalable transport SHOULD set the ECT bit in SYN, SYN/ACK and pure ACK packets.

Motivation: Failing to set the ECT bit in SYN, SYN/ACK or ACK packets results in these packets being more likely dropped during congestion events. Dropping SYN and SYN/ACK packets is particularly bad for performance as the retransmission timers for these packets are large. [RFC3168] prevents from marking these packets due to security reasons. The arguments provided should be revisited in the context of L4S and evaluate if avoiding marking these packets is still the best approach.

Optimization #5.2: Faster than additive increase.

Description: Scalable transport MAY support faster than additive increase in the congestion avoidance phase.

Motivation: As currently defined, DCTCP supports additive increase in congestion avoidance phase. It would be beneficial for performance to update the congestion control algorithm to increase the CWND more than 1 MSS per RTT during the congestion avoidance phase. In the context of L4S such mechanism, must also provide fairness with other classes of traffic, including classic TCP and possibly scalable TCP that uses additive increase.

Optimization #5.3: Faster convergence to fairness.

Description: Scalable transport SHOULD converge to a fair share allocation of the available capacity as fast as classic TCP or faster.

Motivation: The time required for a new flow to obtain its fair share of the capacity of the bottleneck when there are already ongoing flows using up all the bottleneck capacity is higher in the case of DCTCP than in the case of classic TCP (about a factor of 1,5 and 2 larger according to [Alizadeh-stability]). This is detrimental in general, but it is very harmful for short flows, which performance can be worse than the one obtained with classic TCP. For this reason it is desirable that scalable transport provide convergence times no larger than classic TCP.

Appendix B. Standardization items

The following table includes all the items that should be standardized to provide a full L4S architecture.

The table is too wide for the ASCII draft format, so it has been split into two, with a common column of row index numbers on the left.

The columns in the second part of the table have the following meanings:

WG: The IETF WG most relevant to this requirement. The "tcpm/iccrgr" combination refers to the procedure typically used for congestion control changes, where tcpm owns the approval decision, but uses the iccrgr for expert review [NewCC_Proc];

TCP: Applicable to all forms of TCP congestion control;

DCTCP: Applicable to Data Centre TCP as currently used (in controlled environments);

DCTCP bis: Applicable to an future Data Centre TCP congestion control intended for controlled environments;

XXX Prague: Applicable to a Scalable variant of XXX (TCP/SCTP/RMCA) congestion control.

Req #	Requirement	Reference
0	ARCHITECTURE	
1	L4S IDENTIFIER	[I-D.briscoe-tsvwg-ecn-l4s-id]
2	DUAL QUEUE AQM	[I-D.briscoe-aqm-dualq-coupled]
3	Suitable ECN Feedback	[I-D.ietf-tcpm-accurate-ecn], [I-D.stewart-tsvwg-sctpecn].
	SCALABLE TRANSPORT - SAFETY ADDITIONS	
4-1	Fall back to Reno/Cubic on loss	[I-D.ietf-tcpm-dctcp]
4-2	Fall back to Reno/Cubic if classic ECN bottleneck detected	
4-3	Reduce RTT-dependence	
4-4	Scaling TCP's Congestion Window for Small Round Trip Times	[TCP-sub-mss-w]
	SCALABLE TRANSPORT - PERFORMANCE ENHANCEMENTS	
5-1	Setting ECT in SYN, SYN/ACK and pure ACK packets	draft-bagnulo-tsvwg-generalized-ECN
5-2	Faster-than-additive increase	
5-3	Less drastic exit from slow-start	

#	WG	TCP	DCTCP	DCTCP-bis	TCP Prague	SCTP Prague	RMCAT Prague
0	tsvwg?	Y	Y	Y	Y	Y	Y
1	tsvwg?			Y	Y	Y	Y
2	aqm?	n/a	n/a	n/a	n/a	n/a	n/a
3	tcpm	Y	Y	Y	Y	n/a	n/a
4-1	tcpm		Y	Y	Y	Y	Y
4-2	tcpm/ iccrgr?				Y	Y	?
4-3	tcpm/ iccrgr?			Y	Y	Y	?
4-4	tcpm	Y	Y	Y	Y	Y	?
5-1	tsvwg	Y	Y	Y	Y	n/a	n/a
5-2	tcpm/ iccrgr?			Y	Y	Y	?
5-3	tcpm/ iccrgr?			Y	Y	Y	?

Authors' Addresses

Bob Briscoe (editor)
Simula Research Lab

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

Koen De Schepper
Nokia Bell Labs
Antwerp
Belgium

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

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
Spain

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: <http://www.it.uc3m.es>

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

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Scope of RFC 6040	2
2. Terminology	2
3. IP-in-IP Tunnels with Tightly Coupled Shim Headers	2
4. IANA Considerations (to be removed by RFC Editor)	3
5. Security Considerations	4
6. Comments Solicited	4
7. Normative References	4
Author's Address	5

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:

- o L2TPv2 [RFC2661], L2TPv3 [RFC3931]
- o GRE [RFC1701], [RFC2784]
- o PPTP [RFC2637]
- o GTP [GTPv1], [GTPv1-U], [GTPv2-C]
- o 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/>

TSVWG
Internet-Draft
Intended status: Informational
Expires: June 18, 2017

R. Geib, Ed.
Deutsche Telekom
D. Black
Dell EMC
December 15, 2016

Diffserv-Interconnection classes and practice
draft-ietf-tsvwg-diffserv-intercon-14

Abstract

This document defines a limited common set of Diffserv Per Hop Behaviours (PHBs) and codepoints (DSCPs) to be applied at (inter)connections of two separately administered and operated networks, and explains how this approach can simplify network configuration and operation. Many network providers operate Multi Protocol Label Switching (MPLS) using Treatment Aggregates for traffic marked with different Diffserv Per Hop Behaviors, and use MPLS for interconnection with other networks. This document offers a simple interconnection approach that may simplify operation of Diffserv for network interconnection among providers that use MPLS and apply the Short-Pipe tunnel mode. While motivated by the requirements of MPLS network operators that use Short-Pipe tunnels, this document is applicable to other networks, both MPLS and non-MPLS.

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 June 18, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Related work	4
1.2. Applicability Statement	4
1.3. Document Organization	5
2. MPLS and the Short Pipe tunnel model	5
3. Relationship to RFC5127	6
3.1. RFC5127 Background	6
3.2. Differences from RFC5127	7
4. The Diffserv-Intercon Interconnection Classes	8
4.1. Diffserv-Intercon Example	10
4.2. End-to-end PHB and DSCP Transparency	13
4.3. Treatment of Network Control traffic at carrier interconnection interfaces	14
5. Acknowledgements	15
6. IANA Considerations	15
7. Security Considerations	15
8. References	16
8.1. Normative References	16
8.2. Informative References	16
Appendix A. Appendix A The MPLS Short Pipe Model and IP traffic	18
Authors' Addresses	21

1. Introduction

Diffserv has been deployed in many networks; it provides differentiated traffic forwarding based on the Diffserv Codepoint (DSCP) field, which is part of the IP header [RFC2474]. This document defines a set of common Diffserv classes (Per Hop Behaviors, PHBs) and code points for use at interconnection points to which and from which locally used classes and code points should be mapped.

As described by section 2.3.4.2 of RFC2475, remarking of packets at domain boundaries is a Diffserv feature [RFC2475]. If traffic marked with unknown or unexpected DSCPs is received, RFC2474 recommends forwarding that traffic with default (best effort) treatment without changing the DSCP markings to better support incremental Diffserv deployment in existing networks as well as with routers that do not support Diffserv or are not configured to support it. Many networks do not follow this recommendation, and instead remark unknown or unexpected DSCPs to zero upon receipt for default (best effort) forwarding in accordance with the guidance in RFC2475 [RFC2475] to ensure that appropriate DSCPs are used within a Diffserv domain. This draft is based on the latter approach, and defines additional DSCPs that are known and expected at network interconnection interfaces in order to reduce the amount of traffic whose DSCPs are remarked to zero.

This document is motivated by requirements for IP network interconnection with Diffserv support among providers that operate Multi Protocol Label Switching (MPLS) in their backbones, but is also applicable to other technologies. The operational simplifications and methods in this document help align IP Diffserv functionality with MPLS limitations resulting from the widely deployed Short Pipe tunnel model for operation [RFC3270]. Further, limiting Diffserv to a small number of Treatment Aggregates can enable network traffic to leave a network with the DSCP value with which it was received, even if a different DSCP is used within the network, thus providing an opportunity to extend consistent Diffserv treatment across network boundaries.

In isolation, use of a defined set of interconnection PHBs and DSCPs may appear to be additional effort for a network operator. The primary offsetting benefit is that mapping from or to the interconnection PHBs and DSCPs is specified once for all of the interconnections to other networks that can use this approach. Absent this approach, the PHBs and DSCPs have to be negotiated and configured independently for each network interconnection, which has poor administrative and operational scaling properties. Further, consistent end-to-end Diffserv treatment is more likely to result when an interconnection code point scheme is used because traffic is remarked to the same PHBs at all network interconnections.

The interconnection approach described in this document (referred to as Diffserv-Intercon) uses a set of PHBs (mapped to four corresponding MPLS treatment aggregates) along with a set of interconnection DSCPs allowing straightforward rewriting to domain-internal DSCPs and defined DSCP markings for traffic forwarded to interconnected domains. The solution described here can be used in

other contexts benefitting from a defined Diffserv interconnection interface.

The basic idea is that traffic sent with a Diffserv-Interconnect PHB and DSCP is restored to that PHB and DSCP at each network interconnection, even though a different PHB and DSCP may be used by each network involved. The key requirement is that the network ingress interconnect DSCP be restored at network egress, and a key observation is that this is only feasible in general for a small number of DSCPs. Traffic sent with other DSCPs can be remarked to an interconnect DSCP or dealt with via additional agreement(s) among the operators of the interconnected networks; use of the MPLS Short Pipe model favors remarking unexpected DSCPs to zero in the absence of additional agreement(s), as explained further in this document.

In addition to the common interconnecting PHBs and DSCPs, interconnecting operators need to further agree on the tunneling technology used for interconnection (e.g., MPLS, if used) and control or mitigate the impacts of tunneling on reliability and MTU.

1.1. Related work

In addition to the activities that triggered this work, there are additional RFCs and Internet-drafts that may benefit from an interconnection PHB and DSCP scheme. RFC5160 suggests Meta-QoS-Classes to help enabling deployment of standardized end to end QoS classes [RFC5160]. The Diffserv-Intercon class- and codepoint scheme is intended to complement that work (e.g., by enabling a defined set of interconnection DSCPs and PHBs).

Border Gateway Protocol (BGP) signaling Class of Service at interconnection interfaces by BGP [I-D.knoll-idr-cos-interconnect], [ID.ietf-idr-sla] is complementary to Diffserv-Intercon. These two BGP documents focus on exchanging Service Level Agreement (SLA) and traffic conditioning parameters and assume that common PHBs identified by the signaled DSCPs have been established (e.g., via use of the Diffserv-Intercon DSCPs) prior to BGP signaling of PHB id codes.

1.2. Applicability Statement

This document is applicable to use of Differentiated Services for interconnection traffic between networks, and is particularly suited to interconnection of MPLS-based networks that use MPLS Short-pipe tunnels. This document is also applicable to other network technologies, but it is not intended for use within an individual network, where the approach specified in RFC5127 [RFC5127] is among the possible alternatives; see Section 3 for further discussion.

The Diffserv-Intercon approach described in this document simplifies IP based interconnection to domains operating the MPLS Short Pipe model for IP traffic, both terminating within the domain and transiting onward to another domain. Transiting traffic is received and sent with the same PHB and DSCP. Terminating traffic maintains the PHB with which it was received, however the DSCP may change.

Diffserv-Intercon is also applicable to the Pipe tunneling model [RFC2983], [RFC3270], but it is not applicable to the Uniform tunneling model [RFC2983], [RFC3270].

The Diffserv-Intercon approach defines a set of four PHBs for support at interconnections (or network boundaries in general). Corresponding DSCPs for use at an interconnection interface are added. Diffserv-intercon allows for a simple mapping of PHBs and DSCPs to MPLS Treatment Aggregates. It is extensible by IETF standardisation and this allows additional PHBs and DSCPs to be specified for the Diffserv-intercon scheme. Coding space for private interconnection agreements or provider internal services is left too.

1.3. Document Organization

This document is organized as follows: section 2 reviews the MPLS Short Pipe tunnel model for Diffserv Tunnels [RFC3270], because effective support for that model is a crucial goal of Diffserv-Intercon. Section 3 provides background on RFC5127's approach to traffic class aggregation within a Diffserv network domain and contrasts it with the Diffserv-Intercon approach. Section 4 introduces Diffserv-Interconnection Treatment Aggregates, along with the PHBs and DSCPs that they use, and explains how other PHBs (and associated DSCPs) may be mapped to these Treatment Aggregates. Section 4 also discusses treatment of IP traffic, MPLS VPN Diffserv considerations and handling of high-priority Network Management traffic. Appendix A describes how the MPLS Short Pipe model (penultimate hop popping) impacts DSCP marking for IP interconnections.

2. MPLS and the Short Pipe tunnel model

This section provides a summary of the implications of the MPLS Short Pipe tunnels, and in particular their use of Penultimate Hop Popping (PHP, see RFC3270) on the Diffserv tunnel framework described in RFC2983. The Pipe and Uniform models for Differentiated Services and Tunnels are defined in [RFC2983]. RFC3270 adds the Short Pipe model to reflect the impact of MPLS PHP, primarily for MPLS-based IP tunnels and VPNs. The Short Pipe model and PHP have subsequently become popular with network providers that operate MPLS networks and

are now widely used to transport unencapsulated IP traffic. This has important implications for Diffserv functionality in MPLS networks.

RFC2474's recommendation to forward traffic with unrecognized DSCPs with Default (best effort) service without rewriting the DSCP has not been widely deployed in practice. Network operation and management are simplified when there is a 1-1 match between the DSCP marked on the packet and the forwarding treatment (PHB) applied by network nodes. When this is done, CS0 (the all-zero DSCP) is the only DSCP used for Default forwarding of best effort traffic, and a common practice is to remark to CS0 any traffic received with unrecognized or unsupported DSCPs at network edges.

MPLS networks are more subtle in this regard, as it is possible to encode the provider's DSCP in the MPLS Traffic Class (TC) field and allow that to differ from the PHB indicated by the DSCP in the MPLS-encapsulated IP packet. If the MPLS label with the provider's TC field is present at all hops within the provider network, this approach would allow an unrecognized DSCP to be carried edge-to-edge over an MPLS network, because the effective DSCP used by the provider's MPLS network would be encoded in the MPLS label TC field (and also carried edge-to-edge). Unfortunately this is only true for the Pipe tunnel model.

The Short Pipe tunnel model and PHP behave differently because PHP removes and discards the MPLS provider label carrying the provider's TC field before the traffic exits the provider's network. That discard occurs one hop upstream of the MPLS tunnel endpoint (which is usually at the network edge), resulting in no provider TC info being available at tunnel egress. To ensure consistent handling of traffic at the tunnel egress, the DSCP field in the MPLS-encapsulated IP header has to contain a DSCP that is valid for the provider's network, so that IP header cannot be used to carry a different DSCP edge-to-edge. See Appendix A for a more detailed discussion.

3. Relationship to RFC5127

This document draws heavily upon RFC5127's approach to aggregation of Diffserv traffic classes for use within a network, but there are important differences caused by characteristics of network interconnects that differ from links within a network.

3.1. RFC5127 Background

Many providers operate MPLS-based backbones that employ backbone traffic engineering to ensure that if a major link, switch, or router fails, the result will be a routed network that continues to function. Based on that foundation, [RFC5127] introduced the concept

of Diffserv Treatment Aggregates, which enable traffic marked with multiple DSCPs to be forwarded in a single MPLS Traffic Class (TC) based on robust provider backbone traffic engineering. This enables differentiated forwarding behaviors within a domain in a fashion that does not consume a large number of MPLS Traffic Classes.

RFC5127 provides an example aggregation of Diffserv service classes into 4 Treatment Aggregates. A small number of aggregates are used because:

- o The available coding space for carrying Traffic Class information (e.g., Diffserv PHB) in MPLS (and Ethernet) is only 3 bits in size, and is intended for more than just Diffserv purposes (see, e.g., [RFC5129]).
- o The common interconnection DSCPs ought not to use all 8 possible values. This leaves space for future standards, for private bilateral agreements and for local use PHBs and DSCPs.
- o Migrations from one Diffserv code point scheme to a different one is another possible application of otherwise unused DSCPs.

3.2. Differences from RFC5127

Like RFC5127, this document also uses four traffic aggregates, but differs from RFC5127 in some important ways:

- o It follows RFC2475 in allowing the DSCPs used within a network to differ from those to exchange traffic with other networks (at network edges), but provides support to restore ingress DSCP values if one of the recommended interconnect DSCPs in this draft is used. This results in DSCP remarking at both network ingress and network egress, and this draft assumes that such remarking at network edges is possible for all interface types.
- o Diffserv-Intercon suggests limiting the number of interconnection PHBs per Treatment Aggregate to the minimum required. As further discussed below, the number of PHBs per Treatment Aggregate is no more than two. When two PHBs are specified for a Diffserv-Intercon treatment aggregate, the expectation is that the provider network supports DSCPs for both PHBs, but uses a single MPLS TC for the Treatment Aggregate that contains the two PHBs.
- o Diffserv-Intercon suggests mapping other PHBs and DSCPs into the interconnection Treatment Aggregates as further discussed below.
- o Diffserv-Intercon treats network control traffic as a special case. Within a provider's network, the CS6 DSCP is used for local

network control traffic (routing protocols and Operations, Administration, and Maintenance (OAM) traffic that is essential to network operation administration, control and management) that may be destined for any node within the network. In contrast, network control traffic exchanged between networks (e.g., BGP) usually terminates at or close to a network edge, and is not forwarded through the network because it is not part of internal routing or OAM for the receiving network. In addition, such traffic is unlikely to be covered by standard interconnection agreements; rather, it is more likely to be specifically configured (e.g., most networks impose restrictions on use of BGP with other networks for obvious reasons). See Section 4.2 for further discussion.

- o Because RFC5127 used a Treatment Aggregate for network control traffic, Diffserv-Intercon can instead define a fourth traffic aggregate to be defined for use at network interconnections instead of the Network Control aggregate in RFC5127. Network Control traffic may still be exchanged across network interconnections as further discussed in Section 4.2. Diffserv-Intercon uses this fourth traffic aggregate for VoIP traffic, where network-provided service differentiation is crucial, as even minor glitches are immediately apparent to the humans involved in the conversation.

4. The Diffserv-Intercon Interconnection Classes

At an interconnection, the networks involved need to agree on the PHBs used for interconnection and the specific DSCP for each PHB. This document defines a set of 4 interconnection Treatment Aggregates with well-defined DSCPs to be aggregated by them. A sending party remarks DSCPs from internal schemes to the interconnection code points. The receiving party remarks DSCPs to their internal scheme. The interconnect SLA defines the set of DSCPs and PHBs supported across the two interconnected domains and the treatment of PHBs and DSCPs that are not recognized by the receiving domain.

Similar approaches that use a small number of traffic aggregates (including recognition of the importance of VoIP traffic) have been taken in related standards and recommendations from outside the IETF, e.g., Y.1566 [Y.1566], GSMA IR.34 [IR.34] and MEF23.1 [MEF23.1].

The list of the four Diffserv-Interconnect traffic aggregates follows, highlighting differences from RFC5127 and suggesting mappings for all RFC4594 traffic classes to Diffserv-Intercon Treatment Aggregates:

Telephony Service Treatment Aggregate: PHB EF, DSCP 101 110 and PHB VOICE-ADMIT, DSCP 101 100, see [RFC3246], [RFC4594] and [RFC5865]. This Treatment Aggregate corresponds to RFC5127's real time Treatment Aggregate definition regarding the queuing (both delay and jitter should be minimized), but this aggregate is restricted to transport Telephony Service Class traffic in the sense of RFC4594 [RFC4594].

Bulk Real-Time Treatment Aggregate: This Treatment Aggregate is designed to transport PHB AF41, DSCP 100 010 (the other AF4 PHB group PHBs and DSCPs may be used for future extension of the set of DSCPs carried by this Treatment Aggregate). This Treatment Aggregate is intended for Diffserv-Intercon network interconnection of the portions of RFC5127's Real Time Treatment Aggregate, that consume significant bandwidth. This traffic is expected to consist of the RFC4594 classes Broadcast Video, Real-Time Interactive and Multimedia Conferencing. This treatment aggregate should be configured with a rate queue (consistent with RFC4594's recommendation for the transported traffic classes). By comparison to RFC5127, the number of DSCPs has been reduced to one (initially). The AF42 and AF43 PHBs could be added if there is a need for three-color marked Multimedia Conferencing traffic.

Assured Elastic Treatment Aggregate This Treatment Aggregate consists of PHBs AF31 and AF32 (i.e., DSCPs 011 010 and 011 100). By comparison to RFC5127, the number of DSCPs has been reduced to two. This document suggests to transport signaling marked by AF31 (e.g., as recommended by GSMA IR.34 [IR.34]). AF33 is reserved for extension of PHBs to be aggregated by this TA. For Diffserv-Intercon network interconnection, the following RFC4594 service classes should be mapped to the Assured Elastic Treatment Aggregate: the Signaling Service Class (being marked for lowest loss probability), Multimedia Streaming Service Class, the Low-Latency Data Service Class and the High-Throughput Data Service Class.

Default / Elastic Treatment Aggregate: transports the default PHB, CS0 with DSCP 000 000. RFC5127 example refers to this Treatment Aggregate as Aggregate Elastic. An important difference from RFC5127 is that any traffic with unrecognized or unsupported DSCPs may be remarked to this DSCP. For Diffserv-Intercon network interconnection, the RFC4594 standard service class and Low-priority Data service class should be mapped to this Treatment Aggregate. This document does not specify an interconnection class for RFC4594 Low-

priority data. This data may be forwarded by a Lower Effort PHB in one domain (like the PHB proposed by Informational [RFC3662]), but using the methods specified in this document will be remarked with DSCP CS0 at a Diffserv-Intercon network interconnection. This has the effect that Low-priority data is treated the same as data sent using the default class. (Note: In a network that implements RFC2474, Low-priority traffic marked as CS1 would otherwise receive better treatment than traffic using the default class.)

RFC2475 states that Ingress nodes must condition all inbound traffic to ensure that the DS codepoints are acceptable; packets found to have unacceptable codepoints must either be discarded or must have their DS codepoints modified to acceptable values before being forwarded. For example, an ingress node receiving traffic from a domain with which no enhanced service agreement exists may reset the DS codepoint to CS0. As a consequence, an interconnect SLA needs to specify not only the treatment of traffic that arrives with a supported interconnect DSCP, but also the treatment of traffic that arrives with unsupported or unexpected DSCPs; remarking to CS0 is a widely deployed behavior.

During the process of setting up a Diffserv interconnection, both networks should define the set of acceptable and unacceptable DSCPs and specify the treatment of traffic marked with each DSCP.

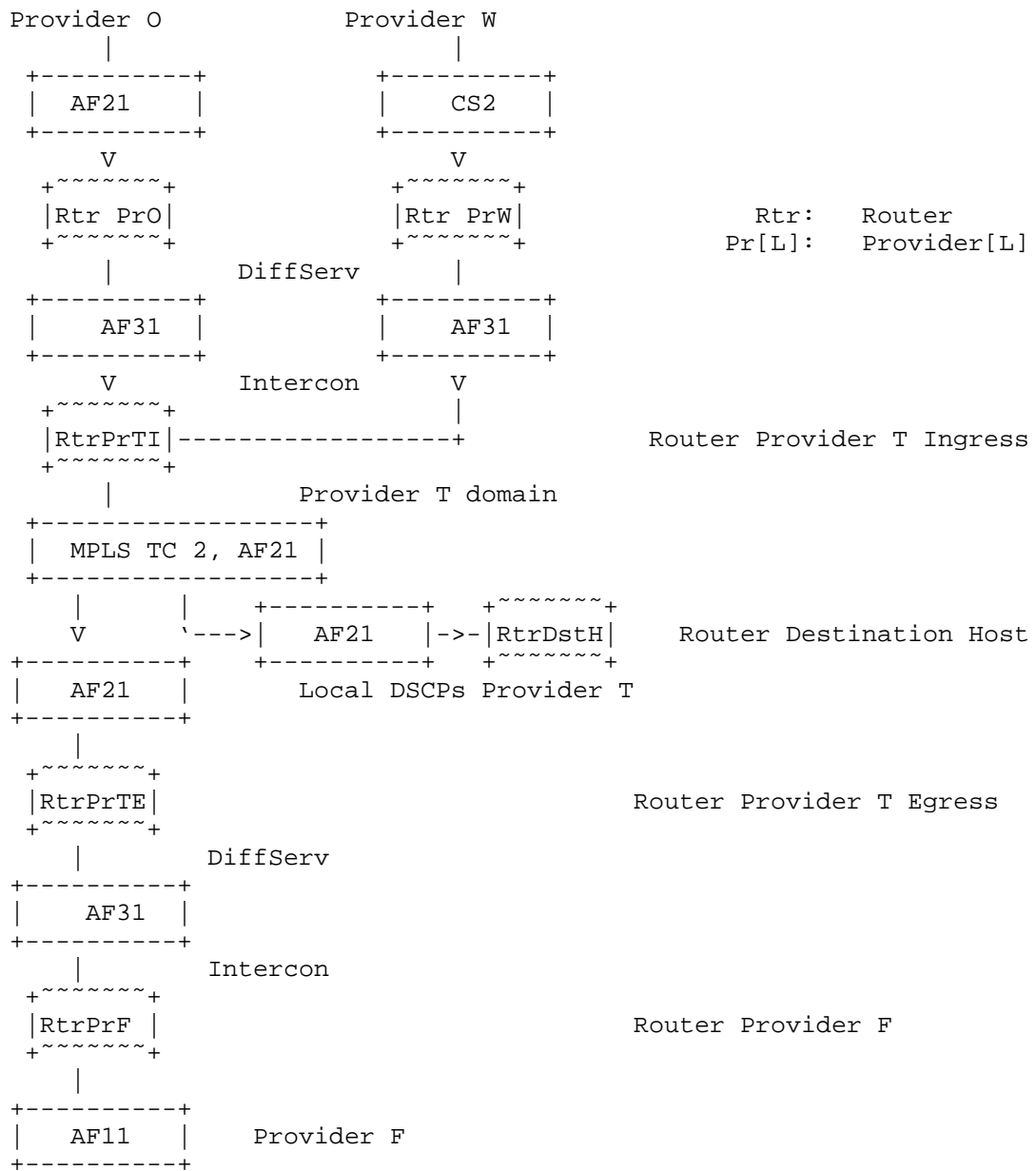
While Diffserv-Intercon allows modification of unacceptable DSCPs, if traffic using one or more of the PHBs in a PHB group (e.g., AF3x, consisting of AF31, AF32 and AF33) is accepted as part of a supported Diffserv-Intercon Treatment Aggregate, then traffic using other PHBs from the same PHB group should not be modified to use PHBs outside of that PHB group, and in particular should not be remarked to CS0 unless the entire PHB group is remarked to CS0. This avoids unexpected forwarding behavior (and potential reordering, see also [RFC7657]) when using Assured Forwarding (AF) PHBs [RFC2597].

4.1. Diffserv-Intercon Example

The overall approach to DSCP marking at network interconnections is illustrated by the following example. Provider O and provider W are peered with provider T. They have agreed upon a Diffserv interconnection SLA.

Traffic of provider O terminates within provider T's network, while provider W's traffic transits through the network of provider T to provider F. This example assumes that all providers use their own internal PHB and codepoint (DSCP) that correspond to the AF31 PHB in

the Diffserv-Intercon Assured Elastic Treatment Aggregate (AF21 and CS2 are used in the example).



Diffserv-Intercon example

Figure 1

Providers only need to deploy mappings of internal DSCPs to/from Diffserv-Intercon DSCPs so that they can exchange traffic using the desired PHBs. In the example, provider O has decided that the properties of his internal class AF21 are best met by the Diffserv-Intercon Assured Elastic Treatment Aggregate, PHB AF31. At the outgoing peering interface connecting provider O with provider T the former's peering router remarks AF21 traffic to AF31. The domain internal PHB of provider T meeting the requirement of Diffserv-Intercon Assured Elastic Treatment Aggregate are from AF2x PHB group. Hence AF31 traffic received at the interconnection with provider T is remarked to AF21 by the peering router of domain T, and domain T has chosen to use MPLS Traffic Class value 2 for this aggregate. At the penultimate MPLS node, the top MPLS label is removed and exposes the IP header marked by the DSCP that has been set at the network ingress. The peering router connecting domain T with domain F classifies the packet by its domain-T-internal DSCP AF21. As the packet leaves domain T on the interface to domain F, this causes the packet's DSCP to be remarked to AF31. The peering router of domain F classifies the packet for domain-F-internal PHB AF11, as this is the PHB with properties matching Diffserv-Intercon's Assured Elastic Treatment Aggregate.

This example can be extended. The figure shows Provider-W using CS2 for traffic that corresponds to Diffserv-Intercon Assured Elastic Treatment Aggregate PHB AF31; that traffic is mapped to AF31 at the Diffserv-Intercon interconnection to Provider-T. In addition, suppose that Provider-O supports a PHB marked by AF22 and this PHB is supposed to obtain Diffserv transport within Provider-T domain. Then Provider-O will remark it with DSCP AF32 for interconnection to Provider-T.

Finally suppose that Provider-W supports CS3 for internal use only. Then no Diffserv-Intercon DSCP mapping needs to be configured at the peering router. Traffic, sent by Provider-W to Provider-T marked by CS3 due to a misconfiguration may be remarked to CS0 by Provider-T.

4.2. End-to-end PHB and DSCP Transparency

This section briefly discusses end-to-end Diffserv approaches related to the Uniform, Pipe and Short Pipe tunnel models ([RFC2983], [RFC3270]), when used edge-to-edge in a network.

- o With the Uniform model, neither the DSCP nor the PHB change. This implies that a network management packet received with a CS6 DSCP would be forwarded with an MPLS Traffic Class corresponding to CS6. The uniform model is outside the scope of this document.

- o With the Pipe model, the inner tunnel DSCP remains unchanged, but an outer tunnel DSCP and the PHB could be changed. For example a packet received with a (network specific) CS1 DSCP would be transported by default PHB and if MPLS is applicable, forwarded with an MPLS Traffic Class corresponding to Default PHB. The CS1 DSCP is not rewritten. Transport of a large variety (much greater than 4) DSCPs may be required across an interconnected network operating MPLS Short pipe transport for IP traffic. In that case, a tunnel based on the Pipe model is among the possible approaches. The Pipe model is outside the scope of this document.
- o With the Short Pipe model, the DSCP likely changes and the PHB might change. This document describes a method to simplify Diffserv network interconnection when a DSCP rewrite can't be avoided.

4.3. Treatment of Network Control traffic at carrier interconnection interfaces

As specified by RFC4594, section 3.2, Network Control (NC) traffic marked by CS6 is expected at some interconnection interfaces. This document does not change RFC4594, but observes that network control traffic received at network ingress is generally different from network control traffic within a network that is the primary use of CS6 envisioned by RFC4594. A specific example is that some CS6 traffic exchanged across carrier interconnections is terminated at the network ingress node, e.g., when BGP is used between the two routers on opposite ends of an interconnection link; in this case the operators would enter into a bilateral agreement to use CS6 for that BGP traffic.

The end-to-end discussion in the previous section (4.2) is generally inapplicable to network control traffic - network control traffic is generally intended to control a network, not be transported between networks. One exception is that network control traffic makes sense for a purchased transit agreement, and preservation of the CS6 DSCP marking for network control traffic that is transited is reasonable in some cases, although it is generally inappropriate to use CS6 for forwarding that traffic within the network that provides transit. Use of an IP tunnel is suggested in order to conceal the CS6 markings on transiting network control traffic from the network that provides the transit. In this case, Pipe model for Diffserv tunneling is used.

If the MPLS Short Pipe model is deployed for unencapsulated IPv4 traffic, an IP network provider should limit access to the CS6 and CS7 DSCPs so that they are only used for network control traffic for the provider's own network.

Interconnecting carriers should specify treatment of CS6 marked traffic received at a carrier interconnection which is to be forwarded beyond the ingress node. An SLA covering the following cases is recommended when a provider wishes to send CS6 marked traffic across an interconnection link and that traffic's destination is beyond the interconnected ingress node:

- o classification of traffic that is network control traffic for both domains. This traffic should be classified and marked for the CS6 DSCP.
- o classification of traffic that is network control traffic for the sending domain only. This traffic should be forwarded with a PHB that is appropriate for the NC service class [RFC4594], e.g., AF31 as specified by this document. As an example GSMA IR.34 recommends an Interactive class / AF31 to carry SIP and DIAMETER traffic. While this is service control traffic of high importance to interconnected Mobile Network Operators, it is certainly not Network Control traffic for a fixed network providing transit among such operators, and hence should not receive CS6 treatment in such a transit network.
- o any other CS6 marked traffic should be remarked or dropped.

5. Acknowledgements

Bob Briscoe and Gorrry Fairhurst reviewed the draft and provided rich feedback. Brian Carpenter, Fred Baker, Al Morton and Sebastien Jobert discussed the draft and helped improving it. Mohamed Boucadair and Thomas Knoll helped adding awareness of related work. James Polk's discussion during IETF 89 helped to improve the text on the relation of this draft to RFC4594 and RFC5127.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

The DSCP field in the IP header can expose additional traffic classification information at network interconnections by comparison to use of a zero DSCP for all interconnect traffic. If traffic classification info is sensitive, the DSCP field could be remarked to zero to hide the classification as a countermeasure, at the cost of loss of Diffserv info and differentiated traffic handling on the interconnect and subsequent networks. When AF PHBs are used, any such remarking should respect AF PHB group boundaries as further discussed at the end of Section 4.

This document does not introduce new features; it describes how to use existing ones. The Diffserv security considerations in [RFC2475] and [RFC4594] apply.

8. References

8.1. Normative References

- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<http://www.rfc-editor.org/info/rfc2474>>.
- [RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, DOI 10.17487/RFC2597, June 1999, <<http://www.rfc-editor.org/info/rfc2597>>.
- [RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, DOI 10.17487/RFC3246, March 2002, <<http://www.rfc-editor.org/info/rfc3246>>.
- [RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <<http://www.rfc-editor.org/info/rfc3270>>.
- [RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <<http://www.rfc-editor.org/info/rfc5129>>.
- [RFC5865] Baker, F., Polk, J., and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, DOI 10.17487/RFC5865, May 2010, <<http://www.rfc-editor.org/info/rfc5865>>.

8.2. Informative References

- [I-D.knoll-idr-cos-interconnect] Knoll, T., "BGP Class of Service Interconnection", draft-knoll-idr-cos-interconnect-17 (work in progress), November 2016.

- [ID.ietf-idr-sla] IETF, "Inter-domain SLA Exchange", IETF, <http://datatracker.ietf.org/doc/draft-ietf-idr-sla-exchange/>, 2013.
- [IR.34] GSMA Association, "IR.34 Inter-Service Provider IP Backbone Guidelines Version 7.0", GSMA, GSMA IR.34 <http://www.gsma.com/newsroom/wp-content/uploads/2012/03/ir.34.pdf>, 2012.
- [MEF23.1] MEF, "Implementation Agreement MEF 23.1 Carrier Ethernet Class of Service Phase 2", MEF, MEF23.1 http://metroethernetforum.org/PDF_Documents/technical-specifications/MEF_23.1.pdf, 2012.
- [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <<http://www.rfc-editor.org/info/rfc2475>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<http://www.rfc-editor.org/info/rfc2983>>.
- [RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services", RFC 3662, DOI 10.17487/RFC3662, December 2003, <<http://www.rfc-editor.org/info/rfc3662>>.
- [RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <<http://www.rfc-editor.org/info/rfc4594>>.
- [RFC5127] Chan, K., Babiarz, J., and F. Baker, "Aggregation of Diffserv Service Classes", RFC 5127, DOI 10.17487/RFC5127, February 2008, <<http://www.rfc-editor.org/info/rfc5127>>.
- [RFC5160] Levis, P. and M. Boucadair, "Considerations of Provider-to-Provider Agreements for Internet-Scale Quality of Service (QoS)", RFC 5160, DOI 10.17487/RFC5160, March 2008, <<http://www.rfc-editor.org/info/rfc5160>>.
- [RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <<http://www.rfc-editor.org/info/rfc7657>>.

- [Y.1566] ITU-T, "Quality of service mapping and interconnection between Ethernet, IP and multiprotocol label switching networks", ITU, <http://www.itu.int/rec/T-REC-Y.1566-201207-I/en>, 2012.

Appendix A. Appendix A The MPLS Short Pipe Model and IP traffic

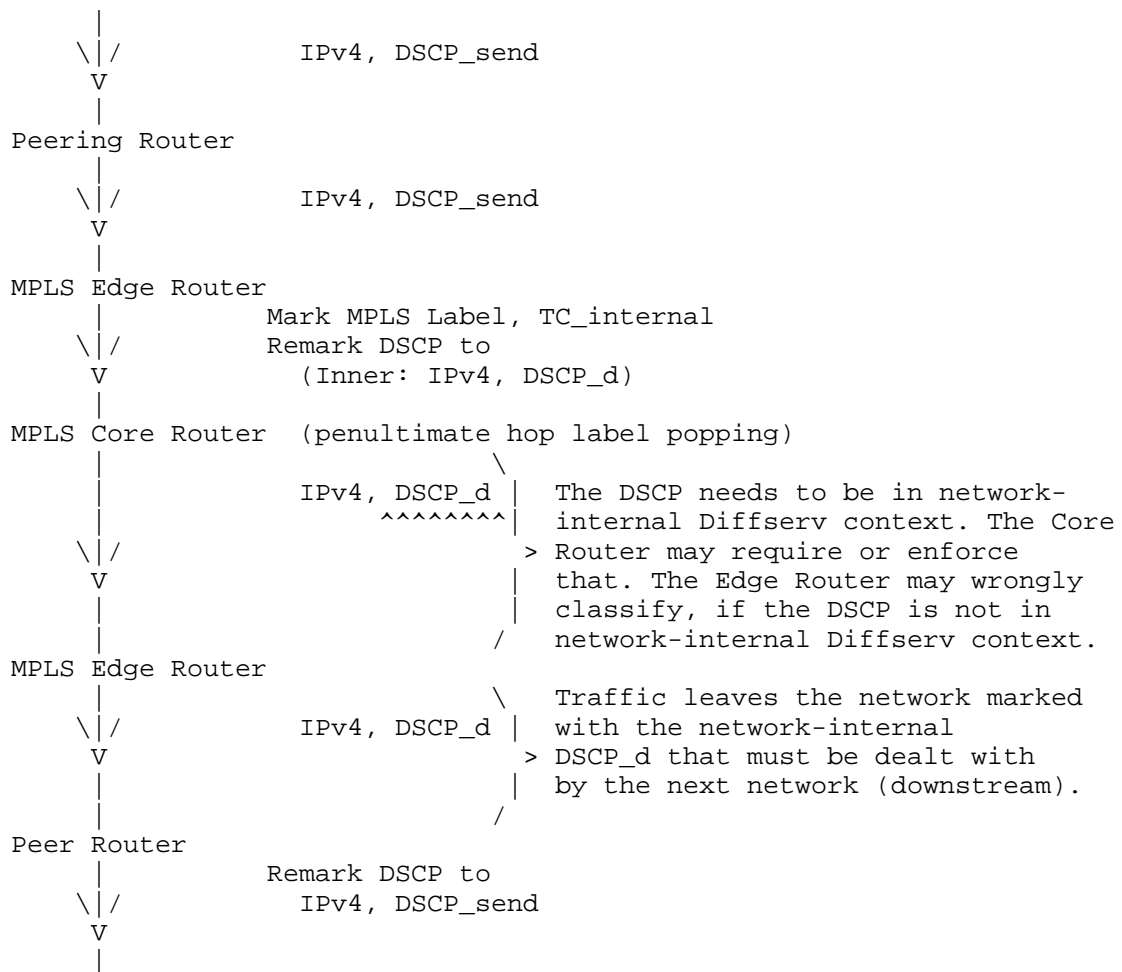
The MPLS Short Pipe Model (or penultimate Hop Label Popping) is widely deployed in carrier networks. If unencapsulated IP traffic is transported using MPLS Short Pipe, IP headers appear inside the last section of the MPLS domain. This impacts the number of PHBs and DSCPs that a network provider can reasonably support. See Figure 2 (below) for an example.

For encapsulated IP traffic, only the outer tunnel header is relevant for forwarding. If the tunnel does not terminate within the MPLS network section, only the outer tunnel DSCP is involved, as the inner DSCP does not affect forwarding behavior; in this case all DSCPs could be used in the inner IP header without affecting network behavior based on the outer MPLS header. Here the Pipe model applies.

Layer 2 and Layer 3 VPN traffic all use an additional MPLS label; in this case, the MPLS tunnel follows the Pipe model. Classification and queuing within an MPLS network is always based on an MPLS label, as opposed to the outer IP header.

Carriers often select PHBs and DSCP without regard to interconnection. As a result PHBs and DSCPs typically differ between network carriers. With the exception of best effort traffic, a DSCP change should be expected at an interconnection at least for unencapsulated IP traffic, even if the PHB is suitably mapped by the carriers involved.

Although RFC3270 suggests that the Short Pipe Model is only applicable to VPNs, current networks also use it to transport non-tunneled IPv4 traffic. This is shown in figure 2 where Diffserv-Intercon is not used, resulting in exposure of the internal DSCPs of the upstream network to the downstream network across the interconnection.



Short-Pipe / penultimate hop popping example

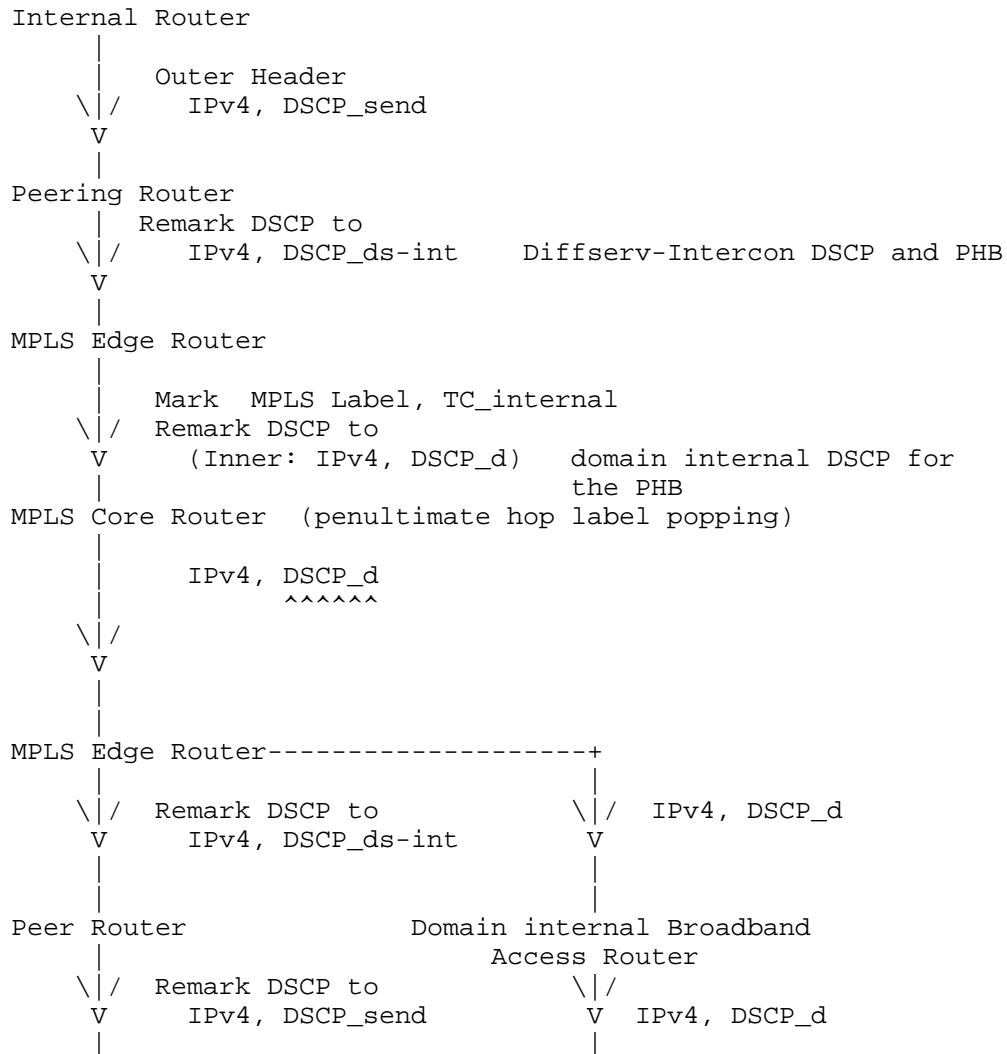
Figure 2

The packets IP DSCP must be in a well understood Diffserv context for schedulers and classifiers on the interfaces of the ultimate MPLS link (last link traversed before leaving the network). The necessary Diffserv context is network-internal and a network operating in this mode enforces DSCP usage in order to obtain robust differentiated forwarding behavior.

Without Diffserv-Intercon treatment, the traffic is likely to leave each network marked with network-internal DSCP. DSCP_send in the

figure above has to be remarked into the first network's Diffserv scheme at the ingress MPLS Edge Router, to DSCP_d in the example. For that reason, the traffic leaves this domain marked by the network-internal DSCP_d. This structure requires that every carrier deploys per-peer PHB and DSCP mapping schemes.

If Diffserv-Intercon is applied DSCPs for traffic transiting the domain can be mapped from and remapped to an original DSCP. This is shown in figure 3. Internal traffic may continue to use internal DSCPs (e.g., DSCP_d) and they may also be used between a carrier and its direct customers.



Short-Pipe example with Diffserv-Intercon

Figure 3

Authors' Addresses

Ruediger Geib (editor)
Deutsche Telekom
Heinrich Hertz Str. 3-7
Darmstadt 64295
Germany

Phone: +49 6151 5812747
Email: Ruediger.Geib@telekom.de

David L. Black
Dell EMC
176 South Street
Hopkinton, MA
USA

Phone: +1 (508) 293-7953
Email: david.black@dell.com

Transport Area Working Group
Internet-Draft
Updates: 3819 (if approved)
Intended status: Best Current Practice
Expires: January 9, 2017

B. Briscoe
Simula Research Laboratory
J. Kaippallimalil
Huawei
P. Thaler
Broadcom Corporation
July 8, 2016

Guidelines for Adding Congestion Notification to Protocols that
Encapsulate IP
draft-ietf-tsvwg-ecn-encap-guidelines-07

Abstract

The purpose of this document is to guide the design of congestion notification in any lower layer or tunnelling protocol that encapsulates IP. The aim is for explicit congestion signals to propagate consistently from lower layer protocols into IP. Then the IP internetwork layer can act as a portability layer to carry congestion notification from non-IP-aware congested nodes up to the transport layer (L4). Following these guidelines should assure interworking between new lower layer congestion notification mechanisms, whether specified by the IETF or other standards bodies.

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

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Scope	5
2. Terminology	6
3. Guidelines in All Cases	7
4. Modes of Operation	8
4.1. Feed-Forward-and-Up Mode	8
4.2. Feed-Up-and-Forward Mode	10
4.3. Feed-Backward Mode	11
4.4. Null Mode	13
5. Feed-Forward-and-Up Mode: Guidelines for Adding Congestion Notification	13
5.1. IP-in-IP Tunnels with Tightly Coupled Shim Headers	14
5.2. Wire Protocol Design: Indication of ECN Support	14
5.3. Encapsulation Guidelines	16
5.4. Decapsulation Guidelines	18
5.5. Sequences of Similar Tunnels or Subnets	19
5.6. Reframing and Congestion Markings	20
6. Feed-Up-and-Forward Mode: Guidelines for Adding Congestion Notification	20
7. Feed-Backward Mode: Guidelines for Adding Congestion Notification	22
8. IANA Considerations (to be removed by RFC Editor)	23
9. Security Considerations	23
10. Conclusions	24
11. Acknowledgements	24
12. Comments Solicited	24
13. References	25
13.1. Normative References	25
13.2. Informative References	25
Appendix A. Outstanding Document Issues	30
Appendix B. Changes in This Version (to be removed by RFC Editor)	30
Authors' Addresses	33

1. Introduction

The benefits of Explicit Congestion Notification (ECN) described below can only be fully realised if support for ECN is added to the relevant subnetwork technology, as well as to IP. When a lower layer buffer drops a packet obviously it does not just drop at that layer; the packet disappears from all layers. In contrast, when a lower layer marks a packet with ECN, the marking needs to be explicitly propagated up the layers. The same is true if a buffer marks the outer header of a packet that encapsulates inner tunnelled headers. Forwarding ECN is not as straightforward as other headers because it has to be assumed ECN may be only partially deployed. If an egress at any layer is not ECN-aware, or if the ultimate receiver or sender is not ECN-aware, congestion needs to be indicated by dropping a packet, not marking it.

The purpose of this document is to guide the addition of congestion notification to any subnet technology or tunnelling protocol, so that lower layer equipment can signal congestion explicitly and it will propagate consistently into encapsulated (higher layer) headers, otherwise the signals will not reach their ultimate destination.

ECN is defined in the IP header (v4 and v6) [RFC3168] to allow a resource to notify the onset of queue build-up without having to drop packets, by explicitly marking a proportion of packets with the congestion experienced (CE) codepoint.

Given a suitable marking scheme, ECN removes nearly all congestion loss and it cuts delays for two main reasons:

- o It avoids the delay when recovering from congestion losses, which particularly benefits small flows or real-time flows, making their delivery time predictably short [RFC2884];
- o As ECN is used more widely by end-systems, it will gradually remove the need to configure a degree of delay into buffers before they start to notify congestion (the cause of bufferbloat). This is because drop involves a trade-off between sending a timely signal and trying to avoid impairment, whereas ECN is solely a signal not an impairment, so there is no harm triggering it earlier.

Some lower layer technologies (e.g. MPLS, Ethernet) are used to form subnetworks with IP-aware nodes only at the edges. These networks are often sized so that it is rare for interior queues to overflow. However, until recently this was more due to the inability of TCP to saturate the links. For many years, fixes such as window scaling [RFC1323] proved hard to deploy. And the New Reno variant of TCP has

remained in widespread use despite its inability to scale to high flow rates. However, now that modern operating systems are finally capable of saturating interior links, even the buffers of well-provisioned interior switches will need to signal episodes of queuing.

Propagation of ECN is defined for MPLS [RFC5129], and is being defined for TRILL [RFC7780], [I-D.eastlake-trill-ecn-support], but it remains to be defined for a number of other subnetwork technologies.

Similarly, ECN propagation is yet to be defined for many tunnelling protocols. [RFC6040] defines how ECN should be propagated for IP-in-IP [RFC2003] and IPsec [RFC4301] tunnels, and it is cited by more recent tunnelling protocols, e.g. Generic UDP Encapsulation (GUE) [I-D.ietf-nvo3-gue] and Geneve [I-D.ietf-nvo3-geneve]. However, as Section 9.3 of RFC3168 pointed out, ECN support will need to be defined for other tunnelling protocols, e.g. L2TP [RFC2661], GRE [RFC1701], [RFC2784], PPTP [RFC2637] and GTP [GTPv1], [GTPv1-U], [GTPv2-C], VXLAN [RFC7348].

Incremental deployment is the most delicate aspect when adding support for ECN. The original ECN protocol in IP [RFC3168] was carefully designed so that a congested buffer would not mark a packet (rather than drop it) unless both source and destination hosts were ECN-capable. Otherwise its congestion markings would never be detected and congestion would just build up further. However, to support congestion marking below the IP layer, it is not sufficient to only check that the two end-points support ECN; correct operation also depends on the decapsulator at each subnet egress faithfully propagating congestion notifications to the higher layer. Otherwise, a legacy decapsulator might silently fail to propagate any ECN signals from the outer to the forwarded header. Then the lost signals would never be detected and again congestion would build up further. The guidelines given later require protocol designers to carefully consider incremental deployment, and suggest various safe approaches for different circumstances.

Of course, the IETF does not have standards authority over every link layer protocol. So this document gives guidelines for designing propagation of congestion notification across the interface between IP and protocols that may encapsulate IP (i.e. that can be layered beneath IP). Each lower layer technology will exhibit different issues and compromises, so the IETF or the relevant standards body must be free to define the specifics of each lower layer congestion notification scheme. Nonetheless, if the guidelines are followed, congestion notification should interwork between different technologies, using IP in its role as a 'portability layer'.

Therefore, the capitalised term 'SHOULD' or 'SHOULD NOT' are often used in preference to 'MUST' or 'MUST NOT', because it is difficult to know the compromises that will be necessary in each protocol design. If a particular protocol design chooses to contradict a 'SHOULD (NOT)' given in the advice below, it MUST include a sound justification.

It has not been possible to give common guidelines for all lower layer technologies, because they do not all fit a common pattern. Instead they have been divided into a few distinct modes of operation: feed-forward-and-upward; feed-upward-and-forward; feed-backward; and null mode. These modes are described in Section 4, then in the following sections separate guidelines are given for each mode.

This document updates the advice to subnetwork designers about ECN in Section 13 of [RFC3819].

1.1. Scope

This document only concerns wire protocol processing of explicit notification of congestion and makes no changes or recommendations concerning algorithms for congestion marking or for congestion response (algorithm issues should be independent of the layer the algorithm operates in).

The question of congestion notification signals with different semantics to those of ECN in IP is touched on in a couple of specific cases (e.g. QCN [IEEE802.1Qau]) and with schemes with multiple severity levels such as PCN [RFC6660]). However, no attempt is made to give guidelines about schemes with different semantics that are yet to be invented.

The semantics of congestion signals can be relative to the traffic class. Therefore correct propagation of congestion signals could depend on correct propagation of any traffic class field between the layers. In this document, correct propagation of traffic class information is assumed, while what 'correct' means and how it is achieved is covered elsewhere (e.g. [RFC2983]) and is outside the scope of the present document.

Note that these guidelines do not require the subnet wire protocol to be changed to accommodate congestion notification. Another way to add congestion notification without consuming header space in the subnet protocol might be to use a parallel control plane protocol.

This document focuses on the congestion notification interface between IP and lower layer protocols that can encapsulate IP, where

the term 'IP' includes v4 or v6, unicast, multicast or anycast. However, it is likely that the guidelines will also be useful when a lower layer protocol or tunnel encapsulates itself (e.g. Ethernet MAC in MAC [IEEE802.1Qah]) or when it encapsulates other protocols. In the feed-backward mode, propagation of congestion signals for multicast and anycast packets is out-of-scope (because it would be so complicated that it is hoped no-one would attempt such an abomination).

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].

Further terminology used within this document:

Protocol data unit (PDU): Information that is delivered as a unit among peer entities of a layered network consisting of protocol control information (typically a header) and possibly user data (payload) of that layer. The scope of this document includes layer 2 and layer 3 networks, where the PDU is respectively termed a frame or a packet (or a cell in ATM). PDU is a general term for any of these. This definition also includes a payload with a shim header lying somewhere between layer 2 and 3.

Transport: The end-to-end transmission control function, conventionally considered at layer-4 in the OSI reference model. Given the audience for this document will often use the word transport to mean low level bit carriage, whenever the term is used it will be qualified, e.g. 'L4 transport'.

Encapsulator: The link or tunnel endpoint function that adds an outer header to a PDU (also termed the 'link ingress', the 'subnet ingress', the 'ingress tunnel endpoint' or just the 'ingress' where the context is clear).

Decapsulator: The link or tunnel endpoint function that removes an outer header from a PDU (also termed the 'link egress', the 'subnet egress', the 'egress tunnel endpoint' or just the 'egress' where the context is clear).

Incoming header: The header of an arriving PDU before encapsulation.

Outer header: The header added to encapsulate a PDU.

Inner header: The header encapsulated by the outer header.

Outgoing header: The header forwarded by the decapsulator.

CE: Congestion Experienced [RFC3168]

ECT: ECN-Capable Transport [RFC3168]

Not-ECT: Not ECN-Capable Transport [RFC3168]

Load Regulator: For each flow of PDUs, the transport function that is capable of controlling the data rate. Typically located at the data source, but in-path nodes can regulate load in some congestion control arrangements (e.g. admission control, policing nodes or transport circuit-breakers [I-D.ietf-tsvwg-circuit-breaker]). Note the term "a function capable of controlling the load" deliberately includes a transport that doesn't actually control the load responsively but ideally it ought to (e.g. a sending application without congestion control that uses UDP).

ECN-PDU: A PDU that is part of a feedback loop within which all the nodes that need to propagate explicit congestion notifications back to the Load Regulator are ECN-capable. An IP packet with a non-zero ECN field implies that the endpoints are ECN-capable, so this would be an ECN-PDU. However, ECN-PDU is intended to be a general term for a PDU at any layer, not just IP.

Not-ECN-PDU: A PDU that is part of a feedback-loop within which some nodes necessary to propagate explicit congestion notifications back to the load regulator are not ECN-capable.

Congestion Baseline: The location of the function on the path that initialised the values of all congestion notification fields in a sequence of packets, before any are set to the congestion experienced (CE) codepoint if they experience congestion further downstream. Typically the original data source at layer-4.

3. Guidelines in All Cases

RFC 3168 specifies that the ECN field in the IP header is intended to be marked by active queue management algorithms. Any congestion notification from an algorithm that does not conform to the recommendations in [RFC7567] MUST NOT be propagated from a lower layer into the ECN field in IP (see also [RFC4774] on alternate uses of the ECN field).

4. Modes of Operation

This section sets down the different modes by which congestion information is passed between the lower layer and the higher one. It acts as a reference framework for the following sections, which give normative guidelines for designers of explicit congestion notification protocols, taking each mode in turn:

Feed-Forward-and-Up: Nodes feed forward congestion notification towards the egress within the lower layer then up and along the layers towards the end-to-end destination at the transport layer. The following local optimisation is possible:

Feed-Up-and-Forward: A lower layer switch feeds-up congestion notification directly into the ECN field in the higher layer (e.g. IP) header, irrespective of whether the node is at the egress of a subnet.

Feed-Backward: Nodes feed back congestion signals towards the ingress of the lower layer and (optionally) attempt to control congestion within their own layer.

Null: Nodes cannot experience congestion at the lower layer except at ingress nodes (which are IP-aware or equivalently higher-layer-aware).

4.1. Feed-Forward-and-Up Mode

Like IP and MPLS, many subnet technologies are based on self-contained protocol data units (PDUs) or frames sent unreliably. They provide no feedback channel at the subnetwork layer, instead relying on higher layers (e.g. TCP) to feed back loss signals.

In these cases, ECN may best be supported by standardising explicit notification of congestion into the lower layer protocol that carries the data forwards. It will then also be necessary to define how the egress of the lower layer subnet propagates this explicit signal into the forwarded upper layer (IP) header. It can then continue forwards until it finally reaches the destination transport (at L4). Then typically the destination will feed this congestion notification back to the source transport using an end-to-end protocol (e.g. TCP). This is the arrangement that has already been used to add ECN to IP-in-IP tunnels [RFC6040], IP-in-MPLS and MPLS-in-MPLS [RFC5129].

This mode is illustrated in Figure 1. Along the middle of the figure, layers 2, 3 and 4 of the protocol stack are shown, and one packet is shown along the bottom as it progresses across the network from source to destination, crossing two subnets connected by a

router, and crossing two switches on the path across each subnet. Congestion at the output of the first switch (shown as *) leads to a congestion marking in the L2 header (shown as C in the illustration of the packet). The chevrons show the progress of the resulting congestion indication. It is propagated from link to link across the subnet in the L2 header, then when the router removes the marked L2 header, it propagates the marking up into the L3 (IP) header. The router forwards the marked L3 header into subnet 2, and when it adds a new L2 header it copies the L3 marking into the L2 header as well, as shown by the 'C's in both layers (assuming the technology of subnet 2 also supports explicit congestion marking).

Note that there is no implication that each 'C' marking is encoded the same; a different encoding might be used for the 'C' marking in each protocol.

Finally, for completeness, we show the L3 marking arriving at the destination, where the host transport protocol (e.g. TCP) feeds it back to the source in the L4 acknowledgement (the 'C' at L4 in the packet at the top of the diagram).

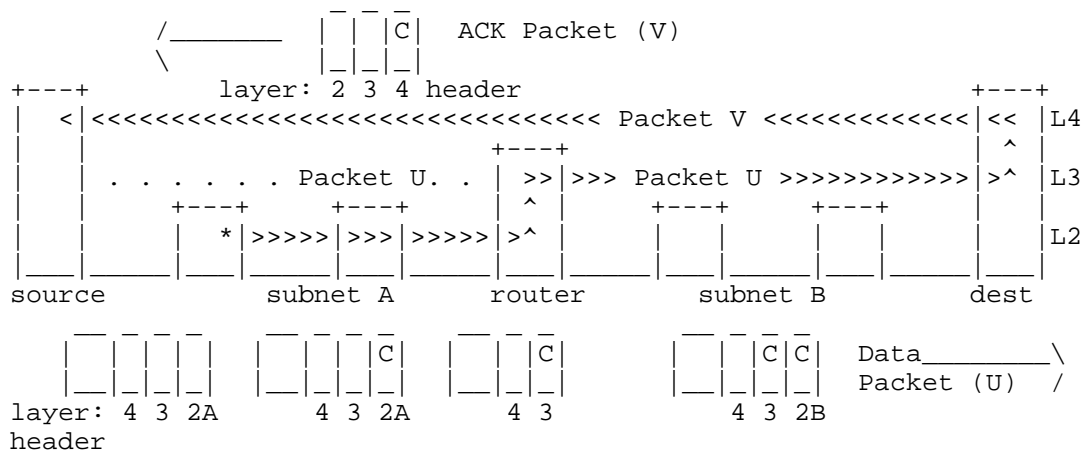


Figure 1: Feed-Forward-and-Up Mode

Of course, modern networks are rarely as simple as this text-book example, often involving multiple nested layers. For example, a 3GPP mobile network may have two IP-in-IP (GTP) tunnels in series and an MPLS backhaul between the base station and the first router. Nonetheless, the example illustrates the general idea of feeding congestion notification forward then upward whenever a header is removed at the egress of a subnet.

Note that the FECN (forward ECN) bit in Frame Relay and the explicit forward congestion indication (EFCI [ITU-T.I.371]) bit in ATM user data cells follow a feed-forward pattern. However, in ATM, this arrangement is only part of a feed-forward-and-backward pattern at the lower layer, not feed-forward-and-up out of the lower layer--the intention was never to interface to IP ECN at the subnet egress. To our knowledge, Frame Relay FECN is solely used to detect where more capacity should be provisioned [Buck00].

4.2. Feed-Up-and-Forward Mode

Ethernet is particularly difficult to extend incrementally to support explicit congestion notification. One way to support ECN in such cases has been to use so called 'layer-3 switches'. These are Ethernet switches that bury into the Ethernet payload to find an IP header and manipulate or act on certain IP fields (specifically Diffserv & ECN). For instance, in Data Center TCP [DCTCP], layer-3 switches are configured to mark the ECN field of the IP header within the Ethernet payload when their output buffer becomes congested. With respect to switching, a layer-3 switch acts solely on the addresses in the Ethernet header; it doesn't use IP addresses, and it doesn't decrement the TTL field in the IP header.

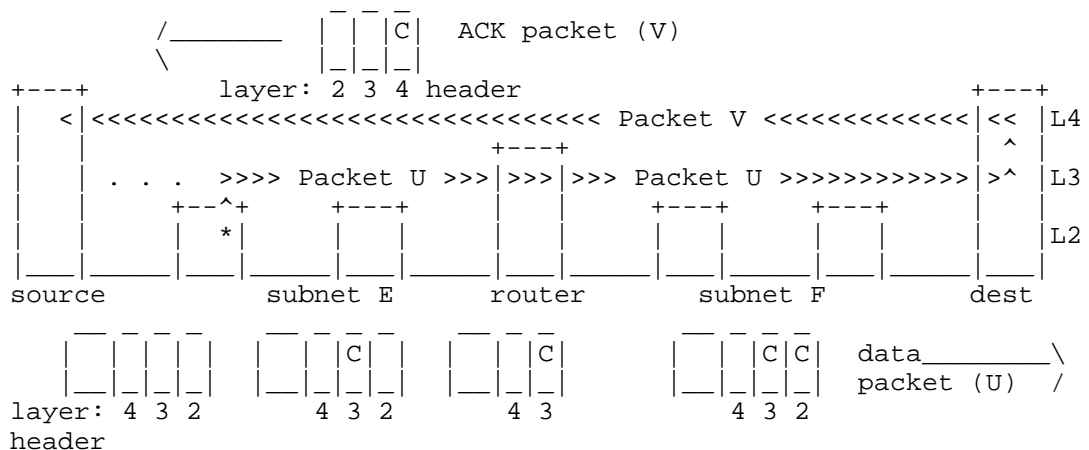


Figure 2: Feed-Up-and-Forward Mode

By comparing Figure 2 with Figure 1, it can be seen that subnet E (perhaps a subnet of layer-3 Ethernet switches) works in feed-up-and-forward mode by notifying congestion directly into L3 at the point of congestion, even though the congested switch does not otherwise act at L3. In this example, the technology in subnet F (e.g. MPLS) does

support ECN natively, so when the router adds the layer-2 header it copies the ECN marking from L3 to L2 as well.

4.3. Feed-Backward Mode

In some layer 2 technologies, explicit congestion notification has been defined for use internally within the subnet with its own feedback and load regulation, but typically the interface with IP for ECN has not been defined.

For instance, for the available bit-rate (ABR) service in ATM, the relative rate mechanism was one of the more popular mechanisms for managing traffic, tending to supersede earlier designs. In this approach ATM switches send special resource management (RM) cells in both the forward and backward directions to control the ingress rate of user data into a virtual circuit. If a switch buffer is approaching congestion or is congested it sends an RM cell back towards the ingress with respectively the No Increase (NI) or Congestion Indication (CI) bit set in its message type field [ATM-TM-ABR]. The ingress then holds or decreases its sending bit-rate accordingly.

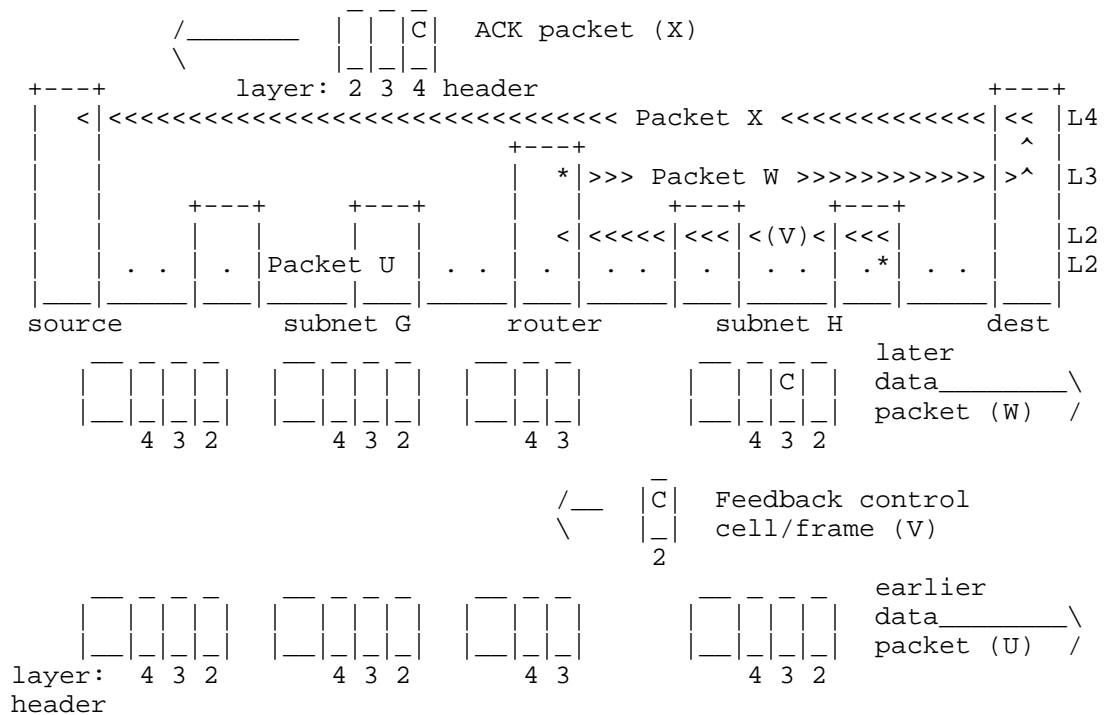


Figure 3: Feed-Backward Mode

ATM's feed-backward approach doesn't fit well when layered beneath IP's feed-forward approach--unless the initial data source is the same node as the ATM ingress. Figure 3 shows the feed-backward approach being used in subnet H. If the final switch on the path is congested (*), it doesn't feed-forward any congestion indications on packet (U). Instead it sends a control cell (V) back to the router at the ATM ingress.

However, the backward feedback doesn't reach the original data source directly because IP doesn't support backward feedback (and subnet G is independent of subnet H). Instead, the router in the middle throttles down its sending rate but the original data sources don't reduce their rates. The resulting rate mismatch causes the middle router's buffer at layer 3 to back up until it becomes congested, which it signals forwards on later data packets at layer 3 (e.g. packet W). Note that the forward signal from the middle router is not triggered directly by the backward signal. Rather, it is triggered by congestion resulting from the middle router's mismatched rate response to the backward signal.

In response to this later forward signalling, end-to-end feedback at layer-4 finally completes the tortuous path of congestion indications back to the origin data source, as before.

4.4. Null Mode

Often link and physical layer resources are 'non-blocking' by design. In these cases congestion notification may be implemented but it does not need to be deployed at the lower layer; ECN in IP would be sufficient.

A degenerate example is a point-to-point Ethernet link. Excess loading of the link merely causes the queue from the higher layer to back up, while the lower layer remains immune to congestion. Even a whole meshed subnetwork can be made immune to interior congestion by limiting ingress capacity and sufficient sizing of interior links, e.g. a non-blocking fat-tree network. An alternative to fat links near the root is numerous thin links with multi-path routing to ensure even worst-case patterns of load cannot congest any link, e.g. a Clos network.

5. Feed-Forward-and-Up Mode: Guidelines for Adding Congestion Notification

Feed-forward-and-up is the mode already used for signalling ECN up the layers through MPLS into IP [RFC5129] and through IP-in-IP tunnels [RFC6040]. These RFCs take a consistent approach and the following guidelines are designed to ensure this consistency continues as ECN support is added to other protocols that encapsulate IP. The guidelines are also designed to ensure compliance with the more general best current practice for the design of alternate ECN schemes given in [RFC4774].

The rest of this section is structured as follows:

- o Section 5.1 addresses the most straightforward cases, where [RFC6040] can be applied directly to add ECN to tunnels that are effectively the same as IP-in-IP tunnels.
- o The subsequent sections give guidelines for adding ECN to a subnet technology that uses feed-forward-and-up mode like IP, but it is not so similar to IP that [RFC6040] rules can be applied directly. Specifically:
 - * Sections 5.2, 5.3 and 5.4 respectively address how to add ECN support to the wire protocol and to the encapsulators and decapsulators at the ingress and egress of the subnet.

- * Section 5.5 deals with the special, but common, case of sequences of tunnels or subnets that all use the same technology
- * Section 5.6 deals with the question of reframing when IP packets do not map 1:1 into lower layer frames.

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

A common pattern for many tunnelling protocols is to encapsulate an inner IP header with shim header(s) then an outer IP header. 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 (such as those listed in the Introduction), the rules in [RFC6040] for propagating the ECN field can be applied directly between the inner and outer IP headers. [I-D.briscoe-tsvwg-rfc6040bis] clarifies that RFC 6040 is just as applicable when there is a tightly-coupled shim between two IP headers as when there is not.

5.2. Wire Protocol Design: Indication of ECN Support

This section is intended to guide the redesign of any lower layer protocol that encapsulate IP to add native ECN support at the lower layer. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

A lower layer (or subnet) congestion notification system:

1. SHOULD NOT apply explicit congestion notifications to PDUs that are destined for legacy layer-4 transport implementations that will not understand ECN, and
2. SHOULD NOT apply explicit congestion notifications to PDUs if the egress of the subnet might not propagate congestion notifications onward into the higher layer.

We use the term ECN-PDUs for a PDU on a feedback loop that will propagate congestion notification properly because it meets both the above criteria. And a Not-ECN-PDU is a PDU on a feedback loop that does not meet both criteria, and will therefore not

propagate congestion notification properly. A corollary of the above is that a lower layer congestion notification protocol:

3. SHOULD be able to distinguish ECN-PDUs from Not-ECN-PDUs.

Note that there is no need for all interior nodes within a subnet to be able to mark congestion explicitly. A mix of ECN and drop signals from different nodes is fine. However, if any interior nodes might generate ECN markings, guideline 2 above says that all relevant egress node(s) SHOULD be able to propagate those markings up to the higher layer.

In IP, if the ECN field in each PDU is cleared to the Not-ECT (not ECN-capable transport) codepoint, it indicates that the L4 transport will not understand congestion markings. A congested buffer must not mark these Not-ECT PDUs, and therefore drops them instead.

The mechanism a lower layer uses to distinguish the ECN-capability of PDUs need not mimic that of IP. The above guidelines merely say that the lower layer system, as a whole, should achieve the same outcome. For instance, ECN-capable feedback loops might use PDUs that are identified by a particular set of labels or tags. Alternatively, logical link protocols that use flow state might determine whether a PDU can be congestion marked by checking for ECN-support in the flow state. Other protocols might depend on out-of-band control signals.

The per-domain checking of ECN support in MPLS [RFC5129] is a good example of a way to avoid sending congestion markings to transports that will not understand them, without using any header space in the subnet protocol.

In MPLS, header space is extremely limited, therefore RFC5129 does not provide a field in the MPLS header to indicate whether the PDU is an ECN-PDU or a Not-ECN-PDU. Instead, interior nodes in a domain are allowed to set explicit congestion indications without checking whether the PDU is destined for a transport that will understand them. Nonetheless, this is made safe by requiring that the network operator upgrades all decapsulating edges of a whole domain at once, as soon as even one switch within the domain is configured to mark rather than drop during congestion. Therefore, any edge node that might decapsulate a packet will be capable of checking whether the higher layer transport is ECN-capable. When decapsulating a CE-marked packet, if the decapsulator discovers that the higher layer (inner header) indicates the transport is not ECN-capable, it drops the packet--effectively on behalf of the earlier congested node (see Decapsulation Guideline 1 in Section 5.4).

It was only appropriate to define such an incremental deployment strategy because MPLS is targeted solely at professional operators, who can be expected to ensure that a whole subnetwork is consistently configured. This strategy might not be appropriate for other link technologies targeted at zero-configuration deployment or deployment by the general public (e.g. Ethernet). For such 'plug-and-play' environments it will be necessary to invent a failsafe approach that ensures congestion markings will never fall into black holes, no matter how inconsistently a system is put together. Alternatively, congestion notification relying on correct system configuration could be confined to flavours of Ethernet intended only for professional network operators, such as IEEE 802.1ah Provider Backbone Bridges (PBB).

ECN support in TRILL [I-D.eastlake-trill-ecn-support] provides a good example of how to add ECN to a lower layer protocol without relying on careful and consistent operator configuration. TRILL provides an extension header word with space for flags of different categories depending on whether logic to understand the extension is critical. The congestion experienced marking has been defined as a 'critical ingress-to-egress' flag. So if a transit RBridge sets this flag and an egress RBridge does not have any logic to process it, it will drop it; which is the desired default action anyway. Therefore TRILL RBridges can be updated with support for ECN in no particular order and, at the egress of the TRILL campus, congestion notification will be propagated to IP as ECN whenever ECN logic has been implemented, and as drop otherwise.

QCN [IEEE802.1Qau] provides another example of how to indicate to lower layer devices that the end-points will not understand ECN. An operator can define certain 802.1p classes of service to indicate non-QCN frames and an ingress bridge is required to map arriving not-QCN-capable IP packets to one of these non-QCN 802.1p classes.

5.3. Encapsulation Guidelines

This section is intended to guide the redesign of any node that encapsulates IP with a lower layer header when adding native ECN support to the lower layer protocol. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

1. Egress Capability Check: A subnet ingress needs to be sure that the corresponding egress of a subnet will propagate any congestion notification added to the outer header across the subnet. This is necessary in addition to checking that an

incoming PDU indicates an ECN-capable (L4) transport. Examples of how this guarantee might be provided include:

- * by configuration (e.g. if any label switches in a domain support ECN marking, [RFC5129] requires all egress nodes to have been configured to propagate ECN)
 - * by the ingress explicitly checking that the egress propagates ECN (e.g. TRILL uses IS-IS to check path capabilities before using critical options [RFC7780])
 - * by inherent design of the protocol (e.g. by encoding ECN marking on the outer header in such a way that a legacy egress that does not understand ECN will consider the PDU corrupt and discard it, thus at least propagating a form of congestion signal).
2. Egress Fails Capability Check: If the ingress cannot guarantee that the egress will propagate congestion notification, the ingress SHOULD disable ECN when it forwards the PDU at the lower layer. An example of how the ingress might disable ECN at the lower layer would be by setting the outer header of the PDU to identify it as a Not-ECN-PDU, assuming the subnet technology supports such a concept.
 3. Standard Congestion Monitoring Baseline: Once the ingress to a subnet has established that the egress will correctly propagate ECN, on encapsulation it SHOULD encode the same level of congestion in outer headers as is arriving in incoming headers. For example it might copy any incoming congestion notification into the outer header of the lower layer protocol.

This ensures that all outer headers reflect congestion accumulated along the whole upstream path since the Load Regulator, not just since the ingress of the subnet. A node that is not the Load Regulator SHOULD NOT re-initialise the level of CE markings in the outer to zero.

This guideline is intended to ensure that any bulk congestion monitoring of outer headers (e.g. by a network management node monitoring ECN in passing frames) is most meaningful. For instance, if an operator measures CE in 0.4% of passing outer headers, this information is only useful if the operator knows where the proportion of CE markings was last initialised to 0% (the Congestion Baseline). Such monitoring information will not be useful if some subnet ingress nodes reset all outer CE markings while others copy incoming CE markings into the outer.

Most information can be extracted if the Congestion Baseline is standardised at the node that is regulating the load (the Load Regulator--typically the data source). Then the operator can measure both congestion since the Load Regulator, and congestion since the subnet ingress. The latter might be measurable by subtracting the level of CE markings on inner headers from that on outer headers (see Appendix C of [RFC6040]).

5.4. Decapsulation Guidelines

This section is intended to guide the redesign of any node that decapsulates IP from within a lower layer header when adding native ECN support to the lower layer protocol. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

A subnet egress SHOULD NOT simply copy congestion notification from outer headers to the forwarded header. It SHOULD calculate the outgoing congestion notification field from the inner and outer headers using the following guidelines. If there is any conflict, rules earlier in the list take precedence over rules later in the list:

1. If the arriving inner header is a Not-ECN-PDU it implies the L4 transport will not understand explicit congestion markings.
Then:
 - * If the outer header carries an explicit congestion marking, drop is the only indication of congestion that the L4 transport will understand. If the congestion marking is the most severe possible, the packet MUST be dropped. However, if congestion can be marked with multiple levels severity and the packet's marking is not the most severe, the packet MAY be forwarded, but it SHOULD be dropped.
 - * If the outer is an ECN-PDU that carries no indication of congestion or a Not-ECN-PDU the PDU SHOULD be forwarded, but still as a Not-ECN-PDU.
2. If the outer header does not support explicit congestion notification (a Not-ECN-PDU), but the inner header does (an ECN-PDU), the inner header SHOULD be forwarded unchanged.
3. In some lower layer protocols congestion may be signalled as a numerical level, such as in the control frames of quantised congestion notification [IEEE802.1Qau]. If such a multi-bit encoding encapsulates an ECN-capable IP data packet, a function

will be needed to convert the quantised congestion level into the frequency of congestion markings in outgoing IP packets.

4. Congestion indications may be encoded by a severity level. For instance increasing levels of congestion might be encoded by numerically increasing indications, e.g. pre-congestion notification (PCN) can be encoded in each PDU at three severity levels in IP or MPLS [RFC6660].

If the arriving inner header is an ECN-PDU, where the inner and outer headers carry indications of congestion of different severity, the more severe indication SHOULD be forwarded in preference to the less severe.

5. The inner and outer headers might carry a combination of congestion notification fields that should not be possible given any currently used protocol transitions. For instance, if Encapsulation Guideline 3 in Section 5.3 had been followed, it should not be possible to have a less severe indication of congestion in the outer than in the inner. It MAY be appropriate to log unexpected combinations of headers and possibly raise an alarm.

If a safe outgoing codepoint can be defined for such a PDU, the PDU SHOULD be forwarded rather than dropped. Some implementers discard PDUs with currently unused combinations of headers just in case they represent an attack. However, an approach using alarms and policy-mediated drop is preferable to hard-coded drop, so that operators can keep track of possible attacks but currently unused combinations are not precluded from future use through new standards actions.

5.5. Sequences of Similar Tunnels or Subnets

In some deployments, particularly in 3GPP networks, an IP packet may traverse two or more IP-in-IP tunnels in sequence that all use identical technology (e.g. GTP).

In such cases, it would be sufficient for every encapsulation and decapsulation in the chain to comply with RFC 6040. Alternatively, as an optimisation, a node that decapsulates a packet and immediately re-encapsulates it for the next tunnel MAY copy the incoming outer ECN field directly to the outgoing outer and the incoming inner ECN field directly to the outgoing inner. Then the overall behavior across the sequence of tunnel segments would still be consistent with RFC 6040.

Appendix C of RFC6040 describes how a tunnel egress can monitor how much congestion has been introduced within a tunnel. A network operator might want to monitor how much congestion had been introduced within a whole sequence of tunnels. Using the technique in Appendix C of RFC6040 at the final egress, the operator could monitor the whole sequence of tunnels, but only if the above optimisation were used consistently along the sequence of tunnels, in order to make it appear as a single tunnel. Therefore, tunnel endpoint implementations SHOULD allow the operator to configure whether this optimisation is enabled.

When ECN support is added to a subnet technology, consideration SHOULD be given to a similar optimisation between subnets in sequence if they all use the same technology.

5.6. Reframing and Congestion Markings

The guidance in this section is worded in terms of framing boundaries, but it applies equally whether the protocol data units are frames, cells or packets.

Where framing boundaries are different between two layers, congestion indications SHOULD be propagated on the basis that a congestion indication on a PDU applies to all the octets in the PDU. On average, an encapsulator or decapsulator SHOULD approximately preserve the number of marked octets arriving and leaving (counting the size of inner headers, but not added encapsulating headers).

The next departing frame SHOULD be immediately marked even if only enough incoming marked octets have arrived for part of the departing frame. This ensures that any outstanding congestion marked octets are propagated immediately, rather than held back waiting for a frame no bigger than the outstanding marked octets--which might involve a long wait.

For instance, an algorithm for marking departing frames could maintain a counter representing the balance of arriving marked octets minus departing marked octets. It adds the size of every marked frame that arrives and if the counter is positive it marks the next frame to depart and subtracts its size from the counter. This will often leave a negative remainder in the counter, which is deliberate.

6. Feed-Up-and-Forward Mode: Guidelines for Adding Congestion Notification

The guidance in this section is applicable, for example, when IP packets:

- o are encapsulated in Ethernet headers, which have no support for ECN;
- o are forwarded by the eNode-B (base station) of a 3GPP radio access network, which is required to apply ECN marking during congestion, [LTE-RA], [UTRAN], but the Packet Data Convergence Protocol (PDCP) that encapsulates the IP header over the radio access has no support for ECN.

This guidance also generalises to encapsulation by other subnet technologies with no native support for explicit congestion notification at the lower layer, but with support for finding and processing an IP header. It is unlikely to be applicable or necessary for IP-in-IP encapsulation, where feed-forward-and-up mode based on [RFC6040] would be more appropriate.

Marking the IP header while switching at layer-2 (by using a layer-3 switch) or while forwarding in a radio access network seems to represent a layering violation. However, it can be considered as a benign optimisation if the guidelines below are followed. Feed-up-and-forward is certainly not a general alternative to implementing feed-forward congestion notification in the lower layer, because:

- o IPv4 and IPv6 are not the only layer-3 protocols that might be encapsulated by lower layer protocols
- o Link-layer encryption might be in use, making the layer-2 payload inaccessible
- o Many Ethernet switches do not have 'layer-3 switch' capabilities so they cannot read or modify an IP payload
- o It might be costly to find an IP header (v4 or v6) when it may be encapsulated by more than one lower layer header, e.g. Ethernet MAC in MAC [IEEE802.1Qah].

Nonetheless, configuring lower layer equipment to look for an ECN field in an encapsulated IP header is a useful optimisation. If the implementation follows the guidelines below, this optimisation does not have to be confined to a controlled environment such as within a data centre; it could usefully be applied on any network--even if the operator is not sure whether the above issues will never apply:

1. If a native lower-layer congestion notification mechanism exists for a subnet technology, it is safe to mix feed-up-and-forward with feed-forward-and-up on other switches in the same subnet. However, it will generally be more efficient to use the native mechanism.

2. The depth of the search for an IP header SHOULD be limited. If an IP header is not found soon enough, or an unrecognised or unreadable header is encountered, the switch SHOULD resort to an alternative means of signalling congestion (e.g. drop, or the native lower layer mechanism if available).
3. It is sufficient to use the first IP header found in the stack; the egress of the relevant tunnel can propagate congestion notification upwards to any more deeply encapsulated IP headers later.

7. Feed-Backward Mode: Guidelines for Adding Congestion Notification

It can be seen from Section 4.3 that congestion notification in a subnet using feed-backward mode has generally not been designed to be directly coupled with IP layer congestion notification. The subnet attempts to minimise congestion internally, and if the incoming load at the ingress exceeds the capacity somewhere through the subnet, the layer 3 buffer into the ingress backs up. Thus, a feed-backward mode subnet is in some sense similar to a null mode subnet, in that there is no need for any direct interaction between the subnet and higher layer congestion notification. Therefore no detailed protocol design guidelines are appropriate. Nonetheless, a more general guideline is appropriate:

A subnetwork technology intended to eventually interface to IP SHOULD NOT be designed using only the feed-backward mode, which is certainly best for a stand-alone subnet, but would need to be modified to work efficiently as part of the wider Internet, because IP uses feed-forward-and-up mode.

The feed-backward approach at least works beneath IP, where the term 'works' is used only in a narrow functional sense because feed-backward can result in very inefficient and sluggish congestion control--except if it is confined to the subnet directly connected to the original data source, when it is faster than feed-forward. It would be valid to design a protocol that could work in feed-backward mode for paths that only cross one subnet, and in feed-forward-and-up mode for paths that cross subnets.

In the early days of TCP/IP, a similar feed-backward approach was tried for explicit congestion signalling, using source-quench (SQ) ICMP control packets. However, SQ fell out of favour and is now formally deprecated [RFC6633]. The main problem was that it is hard for a data source to tell the difference between a spoofed SQ message and a quench request from a genuine buffer on the path. It is also hard for a lower layer buffer to address an SQ message to the

original source port number, which may be buried within many layers of headers, and possibly encrypted.

Quantised congestion notification (QCN--also known as backward congestion notification or BCN) [IEEE802.1Qau] uses a feed-backward mode structurally similar to ATM's relative rate mechanism. However, QCN confines its applicability to scenarios such as some data centres where all endpoints are directly attached by the same Ethernet technology. If a QCN subnet were later connected into a wider IP-based internetwork (e.g. when attempting to interconnect multiple data centres) it would suffer the inefficiency shown Figure 3.

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

This memo includes no request to IANA.

9. Security Considerations

If a lower layer wire protocol is redesigned to include explicit congestion signalling in-band in the protocol header, care SHOULD be taken to ensure that the field used is specified as mutable during transit. Otherwise interior nodes signalling congestion would invalidate any authentication protocol applied to the lower layer header--by altering a header field that had been assumed as immutable.

The redesign of protocols that encapsulate IP in order to propagate congestion signals between layers raises potential signal integrity concerns. Experimental or proposed approaches exist for assuring the end-to-end integrity of in-band congestion signals, e.g.:

- o Congestion exposure (ConEx) for networks to audit that their congestion signals are not being suppressed by other networks or by receivers, and for networks to police that senders are responding sufficiently to the signals, irrespective of the transport protocol used [RFC7713].
- o The ECN nonce [RFC3540] for a TCP sender to detect whether a network or the receiver is suppressing congestion signals.
- o A test with the same goals as the ECN nonce, but without the need for the receiver to co-operate with the protocol [I-D.moncaster-tcpm-rcv-cheat].

Given these end-to-end approaches are already being specified, it would make little sense to attempt to design hop-by-hop congestion signal integrity into a new lower layer protocol, because end-to-end integrity inherently achieves hop-by-hop integrity.

10. Conclusions

Following the guidance in the document enables ECN support to be extended to numerous protocols that encapsulate IP (v4 & v6) in a consistent way, so that IP continues to fulfil its role as an end-to-end interoperability layer. This includes:

- o A wide range of tunnelling protocols with various forms of shim header between two IP headers;
- o A wide range of subnet technologies, particularly those that work in the same 'feed-forward-and-up' mode that is used to support ECN in IP and MPLS.

Guidelines have been defined for supporting propagation of ECN between Ethernet and IP on so-called Layer-3 Ethernet switches, using a 'feed-up-an-forward' mode. This approach could enable other subnet technologies to pass ECN signals into the IP layer, even if they do not support ECN natively.

Finally, attempting to add ECN to a subnet technology in feed-backward mode is deprecated except in special cases, due to its likely sluggish response to congestion.

11. Acknowledgements

Thanks to Gorrry Fairhurst for extensive reviews. Thanks also to the following reviewers: Richard Scheffenegger, Ingemar Johansson, Piers O'Hanlon and Michael Welzl, who pointed out that lower layer congestion notification signals may have different semantics to those in IP. Thanks are also due to the tsvwg chairs, TSV ADs and IETF liaison people such as Eric Gray, Dan Romascanu and Gonzalo Camarillo for helping with the liaisons with the IEEE and 3GPP. And thanks to Georg Mayer and particularly to Erik Guttman for the extensive search and categorisation of any 3GPP specifications that cite ECN specifications.

Bob Briscoe was part-funded by the European Community under its Seventh Framework Programme through the Trilogy project (ICT-216372) for initial drafts and through the Reducing Internet Transport Latency (RITE) project (ICT-317700) subsequently. The views expressed here are solely those of the authors.

12. 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.

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <<http://www.rfc-editor.org/info/rfc5129>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<http://www.rfc-editor.org/info/rfc6040>>.

13.2. Informative References

- [ATM-TM-ABR] Cisco, "Understanding the Available Bit Rate (ABR) Service Category for ATM VCs", Design Technote 10415, June 2005.
- [Buck00] Buckwalter, J., "Frame Relay: Technology and Practice", Pub. Addison Wesley ISBN-13: 978-0201485240, 2000.
- [DCTCP] Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and M. Sridharan, "Data Center TCP (DCTCP)", ACM SIGCOMM CCR 40(4)63--74, October 2010, <<http://portal.acm.org/citation.cfm?id=1851192>>.

- [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.briscoe-tsvwg-rfc6040bis]
Briscoe, B., "Propagating Explicit Congestion Notification Across IP Tunnel Headers Separated by a Shim", draft-briscoe-tsvwg-rfc6040bis-00 (work in progress), July 2016.
- [I-D.eastlake-trill-ecn-support]
3rd, D. and B. Briscoe, "TRILL: ECN (Explicit Congestion Notification) Support", draft-eastlake-trill-ecn-support-00 (work in progress), March 2016.
- [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-circuit-breaker]
Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-15 (work in progress), April 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.
- [IEEE802.1Qah]
IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Virtual Bridged Local Area Networks--Amendment 6: Provider Backbone Bridges", IEEE Std 802.1Qah-2008, August 2008,
<<http://www.ieee802.org/1/pages/802.1ah.html>>.

(Access Controlled link within page)

[IEEE802.1Qau]

Finn, N., Ed., "IEEE Standard for Local and Metropolitan Area Networks--Virtual Bridged Local Area Networks - Amendment 13: Congestion Notification", IEEE Std 802.1Qau-2010, March 2010, <<http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5454061>>.

(Access Controlled link within page)

[ITU-T.I.371]

ITU-T, "Traffic Control and Congestion Control in B-ISDN", ITU-T Rec. I.371 (03/04), March 2004, <<http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5454061>>.

[LTE-RA] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2", Technical Specification TS 36.300.

[RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, DOI 10.17487/RFC1323, May 1992, <<http://www.rfc-editor.org/info/rfc1323>>.

[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>>.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <<http://www.rfc-editor.org/info/rfc2003>>.

[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>>.
- [RFC2884] Hadi Salim, J. and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", RFC 2884, DOI 10.17487/RFC2884, July 2000, <<http://www.rfc-editor.org/info/rfc2884>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<http://www.rfc-editor.org/info/rfc2983>>.
- [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>>.
- [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>>.
- [RFC6633] Gont, F., "Deprecation of ICMP Source Quench Messages", RFC 6633, DOI 10.17487/RFC6633, May 2012, <<http://www.rfc-editor.org/info/rfc6633>>.
- [RFC6660] Briscoe, B., Moncaster, T., and M. Menth, "Encoding Three Pre-Congestion Notification (PCN) States in the IP Header Using a Single Diffserv Codepoint (DSCP)", RFC 6660, DOI 10.17487/RFC6660, July 2012, <<http://www.rfc-editor.org/info/rfc6660>>.
- [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>>.
- [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>>.

- [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>>.
- [RFC7780] Eastlake 3rd, D., Zhang, M., Perlman, R., Banerjee, A., Ghanwani, A., and S. Gupta, "Transparent Interconnection of Lots of Links (TRILL): Clarifications, Corrections, and Updates", RFC 7780, DOI 10.17487/RFC7780, February 2016, <<http://www.rfc-editor.org/info/rfc7780>>.
- [UTRAN] 3GPP, "UTRAN Overall Description", Technical Specification TS 25.401.

Appendix A. Outstanding Document Issues

1. [GF] Concern that certain guidelines warrant a MUST (NOT) rather than a SHOULD (NOT). Given the guidelines say that if any SHOULD (NOT)s are not followed, a strong justification will be needed, they have been left as SHOULD (NOT) pending further list discussion. In particular:

- * If inner is a Not-ECN-PDU and Outer is CE (or highest severity congestion level), MUST (not SHOULD) drop?

This issue has been addressed by explaining when SHOULD or MUST is appropriate.

2. Consider whether an IETF Standard Track doc will be needed to Update the IP-in-IP protocols listed in Section 5.1--at least those that the IETF controls--and which Area it should sit under.

This issue has been addressed by the production of [I-D.briscoe-tsvwg-rfc6040bis], but this text is left outstanding until that draft is adopted.

Appendix B. Changes in This Version (to be removed by RFC Editor)

From ietf-05 to ietf-06:

- * Introduction: Added GUE and Geneve as examples of tightly coupled shims between IP headers that cite RFC 6040. And added VXLAN to list of those that do not.
- * Replaced normative text about tightly coupled shims between IP headers, with reference to new draft-briscoe-tsvwg-rfc6040bis
- * Wire Protocol Design: Indication of ECN Support: Added TRILL as an example of a well-design protocol that does not need an indication of ECN support in the wire protocol.
- * Encapsulation Guidelines: In the case of a Not-ECN-PDU with a CE outer, replaced SHOULD be dropped, with explanations of when SHOULD or MUST are appropriate.
- * Feed-Up-and-Forward Mode: Explained examples more carefully, referred to PDCP and cited UTRAN spec as well as E-UTRAN.
- * Added the people involved in liaisons to the acknowledgements.
- * Updated references.

- * Marked open issues as resolved, but did not delete Open Issues Appendix (yet).

From ietf-04 to ietf-05:

- * Explained why tightly coupled shim headers only "SHOULD" comply with RFC 6040, not "MUST".
- * Updated references

From ietf-03 to ietf-04:

- * Addressed Richard Scheffenegger's review comments: primarily editorial corrections, and addition of examples for clarity.

From ietf-02 to ietf-03:

- * Updated references, ad cited RFC4774.

From ietf-01 to ietf-02:

- * Added Section for guidelines that are applicable in all cases.
- * Updated references.

From ietf-00 to ietf-01: Updated references.

From briscoe-04 to ietf-00: Changed filename following tsvwg adoption.

From briscoe-03 to 04:

- * Re-arranged the introduction to describe the purpose of the document first before introducing ECN in more depth. And clarified the introduction throughout.
- * Added applicability to 3GPP TS 36.300.

From briscoe-02 to 03:

- * Scope section:
 - + Added dependence on correct propagation of traffic class information
 - + For the feed-backward mode, deemed multicast and anycast out of scope

- * Ensured all guidelines referring to subnet technologies also refer to tunnels and vice versa by adding applicability sentences at the start of sections 4.1, 4.2, 4.3, 4.4, 4.6 and 5.
- * Added Security Considerations on ensuring congestion signal fields are classed as immutable and on using end-to-end congestion signal integrity technologies rather than hop-by-hop.

From briscoe-01 to 02:

- * Added authors: JK & PT
- * Added
 - + Section 4.1 "IP-in-IP Tunnels with Tightly Coupled Shim Headers"
 - + Section 4.5 "Sequences of Similar Tunnels or Subnets"
 - + roadmap at the start of Section 4, given the subsections have become quite fragmented.
 - + Section 9 "Conclusions"
- * Clarified why transports are starting to be able to saturate interior links
- * Under Section 1.1, addressed the question of alternative signal semantics and included multicast & anycast.
- * Under Section 3.1, included a 3GPP example.
- * Section 4.2. "Wire Protocol Design":
 - + Altered guideline 2. to make it clear that it only applies to the immediate subnet egress, not later ones
 - + Added a reminder that it is only necessary to check that ECN propagates at the egress, not whether interior nodes mark ECN
 - + Added example of how QCN uses 802.1p to indicate support for QCN.
- * Added references to Appendix C of RFC6040, about monitoring the amount of congestion signals introduced within a tunnel

- * Appendix A: Added more issues to be addressed, including plan to produce a standards track update to IP-in-IP tunnel protocols.
- * Updated acks and references

From briscoe-00 to 01:

- * Intended status: BCP (was Informational) & updates 3819 added.
- * Briefer Introduction: Introductory para justifying benefits of ECN. Moved all but a brief enumeration of modes of operation to their own new section (from both Intro & Scope). Introduced incr. deployment as most tricky part.
- * Tightened & added to terminology section
- * Structured with Modes of Operation, then Guidelines section for each mode.
- * Tightened up guideline text to remove vagueness / passive voice / ambiguity and highlight main guidelines as numbered items.
- * Added Outstanding Document Issues Appendix
- * Updated references

Authors' Addresses

Bob Briscoe
Simula Research Laboratory
UK

EMail: ietf@bobbbriscoe.net
URI: <http://bobbbriscoe.net/>

John Kaippallimalil
Huawei
5340 Legacy Drive, Suite 175
Plano, Texas 75024
USA

EMail: john.kaippallimalil@huawei.com

Pat Thaler
Broadcom Corporation
5025 Keane Drive
Carmichael, CA 95608
USA

EMail: pthaler@broadcom.com

Transport Area Working Group
Internet-Draft
Updates: 3819 (if approved)
Intended status: Best Current Practice
Expires: November 26, 2021

B. Briscoe
Independent
J. Kaippallimalil
Futurewei
May 25, 2021

Guidelines for Adding Congestion Notification to Protocols that
Encapsulate IP
draft-ietf-tsvwg-ecn-encap-guidelines-16

Abstract

The purpose of this document is to guide the design of congestion notification in any lower layer or tunnelling protocol that encapsulates IP. The aim is for explicit congestion signals to propagate consistently from lower layer protocols into IP. Then the IP internetwork layer can act as a portability layer to carry congestion notification from non-IP-aware congested nodes up to the transport layer (L4). Following these guidelines should assure interworking among IP layer and lower layer congestion notification mechanisms, whether specified by the IETF or other standards bodies. This document updates the advice to subnetwork designers about ECN in RFC 3819.

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 November 26, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Update to RFC 3819	5
1.2. Scope	5
2. Terminology	7
3. Modes of Operation	9
3.1. Feed-Forward-and-Up Mode	9
3.2. Feed-Up-and-Forward Mode	11
3.3. Feed-Backward Mode	12
3.4. Null Mode	14
4. Feed-Forward-and-Up Mode: Guidelines for Adding Congestion Notification	14
4.1. IP-in-IP Tunnels with Shim Headers	15
4.2. Wire Protocol Design: Indication of ECN Support	16
4.3. Encapsulation Guidelines	18
4.4. Decapsulation Guidelines	20
4.5. Sequences of Similar Tunnels or Subnets	22
4.6. Reframing and Congestion Markings	22
5. Feed-Up-and-Forward Mode: Guidelines for Adding Congestion Notification	23
6. Feed-Backward Mode: Guidelines for Adding Congestion Notification	24
7. IANA Considerations	25
8. Security Considerations	25
9. Conclusions	26
10. Acknowledgements	27
11. Contributors	27
12. Comments Solicited	27
13. References	27
13.1. Normative References	27
13.2. Informative References	28
Appendix A. Changes in This Version (to be removed by RFC Editor)	33
Authors' Addresses	38

1. Introduction

The benefits of Explicit Congestion Notification (ECN) described in [RFC8087] and summarized below can only be fully realized if support for ECN is added to the relevant subnetwork technology, as well as to IP. When a lower layer buffer drops a packet obviously it does not just drop at that layer; the packet disappears from all layers. In contrast, when active queue management (AQM) at a lower layer marks a packet with ECN, the marking needs to be explicitly propagated up the layers. The same is true if AQM marks the outer header of a packet that encapsulates inner tunnelled headers. Forwarding ECN is not as straightforward as other headers because it has to be assumed ECN may be only partially deployed. If a lower layer header that contains ECN congestion indications is stripped off by a subnet egress that is not ECN-aware, or if the ultimate receiver or sender is not ECN-aware, congestion needs to be indicated by dropping a packet, not marking it.

The purpose of this document is to guide the addition of congestion notification to any subnet technology or tunnelling protocol, so that lower layer AQM algorithms can signal congestion explicitly and it will propagate consistently into encapsulated (higher layer) headers, otherwise the signals will not reach their ultimate destination.

ECN is defined in the IP header (v4 and v6) [RFC3168] to allow a resource to notify the onset of queue build-up without having to drop packets, by explicitly marking a proportion of packets with the congestion experienced (CE) codepoint.

Given a suitable marking scheme, ECN removes nearly all congestion loss and it cuts delays for two main reasons:

- o It avoids the delay when recovering from congestion losses, which particularly benefits small flows or real-time flows, making their delivery time predictably short [RFC2884];
- o As ECN is used more widely by end-systems, it will gradually remove the need to configure a degree of delay into buffers before they start to notify congestion (the cause of bufferbloat). This is because drop involves a trade-off between sending a timely signal and trying to avoid impairment, whereas ECN is solely a signal not an impairment, so there is no harm triggering it earlier.

Some lower layer technologies (e.g. MPLS, Ethernet) are used to form subnetworks with IP-aware nodes only at the edges. These networks are often sized so that it is rare for interior queues to overflow. However, until recently this was more due to the inability of TCP to

saturate the links. For many years, fixes such as window scaling [RFC7323] proved hard to deploy. And the Reno variant of TCP has remained in widespread use despite its inability to scale to high flow rates. However, now that modern operating systems are finally capable of saturating interior links, even the buffers of well-provisioned interior switches will need to signal episodes of queuing.

Propagation of ECN is defined for MPLS [RFC5129], and is being defined for TRILL [RFC7780], [I-D.ietf-trill-ecn-support], but it remains to be defined for a number of other subnetwork technologies.

Similarly, ECN propagation is yet to be defined for many tunnelling protocols. [RFC6040] defines how ECN should be propagated for IP-in-IPv4 [RFC2003], IP-in-IPv6 [RFC2473] and IPsec [RFC4301] tunnels, but there are numerous other tunnelling protocols with a shim and/or a layer 2 header between two IP headers (v4 or v6). Some address ECN propagation between the IP headers, but many do not. This document gives guidance on how to address ECN propagation for future tunnelling protocols, and a companion standards track specification [I-D.ietf-tsvwg-rfc6040update-shim] updates those existing IP-shim-(L2)-IP protocols that are under IETF change control and still widely used.

Incremental deployment is the most delicate aspect when adding support for ECN. The original ECN protocol in IP [RFC3168] was carefully designed so that a congested buffer would not mark a packet (rather than drop it) unless both source and destination hosts were ECN-capable. Otherwise its congestion markings would never be detected and congestion would just build up further. However, to support congestion marking below the IP layer or within tunnels, it is not sufficient to only check that the two layer 4 transport endpoints support ECN; correct operation also depends on the decapsulator at each subnet or tunnel egress faithfully propagating congestion notifications to the higher layer. Otherwise, a legacy decapsulator might silently fail to propagate any ECN signals from the outer to the forwarded header. Then the lost signals would never be detected and again congestion would build up further. The guidelines given later require protocol designers to carefully consider incremental deployment, and suggest various safe approaches for different circumstances.

Of course, the IETF does not have standards authority over every link layer protocol. So this document gives guidelines for designing propagation of congestion notification across the interface between IP and protocols that may encapsulate IP (i.e. that can be layered beneath IP). Each lower layer technology will exhibit different issues and compromises, so the IETF or the relevant standards body

must be free to define the specifics of each lower layer congestion notification scheme. Nonetheless, if the guidelines are followed, congestion notification should interwork between different technologies, using IP in its role as a 'portability layer'.

Therefore, the capitalized terms 'SHOULD' or 'SHOULD NOT' are often used in preference to 'MUST' or 'MUST NOT', because it is difficult to know the compromises that will be necessary in each protocol design. If a particular protocol design chooses not to follow a 'SHOULD (NOT)' given in the advice below, it MUST include a sound justification.

It has not been possible to give common guidelines for all lower layer technologies, because they do not all fit a common pattern. Instead they have been divided into a few distinct modes of operation: feed-forward-and-upward; feed-upward-and-forward; feed-backward; and null mode. These modes are described in Section 3, then in the subsequent sections separate guidelines are given for each mode.

1.1. Update to RFC 3819

This document updates the brief advice to subnetwork designers about ECN in [RFC3819], by replacing the last two paragraphs of Section 13 with the following sentence:

By following the guidelines in [this document], subnetwork designers can enable a layer-2 protocol to participate in congestion control without dropping packets via propagation of explicit congestion notification (ECN [RFC3168]) to receivers.

and adding [this document] as an informative reference. {RFC Editor: Please replace both instances of [this document] above with the number of the present RFC when published.}

1.2. Scope

This document only concerns wire protocol processing of explicit notification of congestion. It makes no changes or recommendations concerning algorithms for congestion marking or for congestion response, because algorithm issues should be independent of the layer the algorithm operates in.

The default ECN semantics are described in [RFC3168] and updated by [RFC8311]. Also the guidelines for AQM designers [RFC7567] clarify the semantics of both drop and ECN signals from AQM algorithms. [RFC4774] is the appropriate best current practice specification of how algorithms with alternative semantics for the ECN field can be

partitioned from Internet traffic that uses the default ECN semantics. There are two main examples for how alternative ECN semantics have been defined in practice:

- o RFC 4774 suggests using the ECN field in combination with a Diffserv codepoint such as in PCN [RFC6660], Voice over 3G [UTRAN] or Voice over LTE (VoLTE) [LTE-RA];
- o RFC 8311 suggests using the ECT(1) codepoint of the ECN field to indicate alternative semantics such as for the experimental Low Latency Low Loss Scalable throughput (L4S) service [I-D.ietf-tsvwg-ecn-l4s-id]).

The aim is that the default rules for encapsulating and decapsulating the ECN field are sufficiently generic that tunnels and subnets will encapsulate and decapsulate packets without regard to how algorithms elsewhere are setting or interpreting the semantics of the ECN field. [RFC6040] updates RFC 4774 to allow alternative encapsulation and decapsulation behaviours to be defined for alternative ECN semantics. However it reinforces the same point - that it is far preferable to try to fit within the common ECN encapsulation and decapsulation behaviours, because expecting all lower layer technologies and tunnels to be updated is likely to be completely impractical.

Alternative semantics for the ECN field can be defined to depend on the traffic class indicated by the DSCP. Therefore correct propagation of congestion signals could depend on correct propagation of the DSCP between the layers and along the path. For instance, if the meaning of the ECN field depends on the DSCP (as in PCN or VoLTE) and if the outer DSCP is stripped on decapsulation, as in the pipe model of [RFC2983], the special semantics of the ECN field would be lost. Similarly, if the DSCP is changed at the boundary between Diffserv domains, the special ECN semantics would also be lost. This is an important implication of the localized scope of most Diffserv arrangements. In this document, correct propagation of traffic class information is assumed, while what 'correct' means and how it is achieved is covered elsewhere (e.g. RFC 2983) and is outside the scope of the present document.

The guidelines in this document do ensure that common encapsulation and decapsulation rules are sufficiently generic to cover cases where ECT(1) is used instead of ECT(0) to identify alternative ECN semantics (as in L4S [I-D.ietf-tsvwg-ecn-l4s-id]) and where ECN marking algorithms use ECT(1) to encode 3 severity levels into the ECN field (e.g. PCN [RFC6660]) rather than the default of 2. All these different semantics for the ECN field work because it has been possible to define common default decapsulation rules that allow for all cases.

Note that the guidelines in this document do not necessarily require the subnet wire protocol to be changed to add support for congestion notification. For instance, the Feed-Up-and-Forward Mode (Section 3.2) and the Null Mode (Section 3.4) do not. Another way to add congestion notification without consuming header space in the subnet protocol might be to use a parallel control plane protocol.

This document focuses on the congestion notification interface between IP and lower layer or tunnel protocols that can encapsulate IP, where the term 'IP' includes v4 or v6, unicast, multicast or anycast. However, it is likely that the guidelines will also be useful when a lower layer protocol or tunnel encapsulates itself, e.g. Ethernet MAC in MAC ([IEEE802.1Q]; previously 802.1ah) or when it encapsulates other protocols. In the feed-backward mode, propagation of congestion signals for multicast and anycast packets is out-of-scope (because the complexity would make it unlikely to be attempted).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Further terminology used within this document:

Protocol data unit (PDU): Information that is delivered as a unit among peer entities of a layered network consisting of protocol control information (typically a header) and possibly user data (payload) of that layer. The scope of this document includes layer 2 and layer 3 networks, where the PDU is respectively termed a frame or a packet (or a cell in ATM). PDU is a general term for any of these. This definition also includes a payload with a shim header lying somewhere between layer 2 and 3.

Transport: The end-to-end transmission control function, conventionally considered at layer-4 in the OSI reference model. Given the audience for this document will often use the word transport to mean low level bit carriage, whenever the term is used it will be qualified, e.g. 'L4 transport'.

Encapsulator: The link or tunnel endpoint function that adds an outer header to a PDU (also termed the 'link ingress', the 'subnet ingress', the 'ingress tunnel endpoint' or just the 'ingress' where the context is clear).

Decapsulator: The link or tunnel endpoint function that removes an outer header from a PDU (also termed the 'link egress', the 'subnet egress', the 'egress tunnel endpoint' or just the 'egress' where the context is clear).

Incoming header: The header of an arriving PDU before encapsulation.

Outer header: The header added to encapsulate a PDU.

Inner header: The header encapsulated by the outer header.

Outgoing header: The header forwarded by the decapsulator.

CE: Congestion Experienced [RFC3168]

ECT: ECN-Capable (L4) Transport [RFC3168]

Not-ECT: Not ECN-Capable (L4) Transport [RFC3168]

Load Regulator: For each flow of PDUs, the transport function that is capable of controlling the data rate. Typically located at the data source, but in-path nodes can regulate load in some congestion control arrangements (e.g. admission control, policing nodes or transport circuit-breakers [RFC8084]). Note the term "a function capable of controlling the load" deliberately includes a transport that does not actually control the load responsively but ideally it ought to (e.g. a sending application without congestion control that uses UDP).

ECN-PDU: A PDU at the IP layer or below with a capacity to signal congestion that is part of a congestion control feedback loop within which all the nodes necessary to propagate the signal back to the Load Regulator are capable of doing that propagation. An IP packet with a non-zero ECN field implies that the endpoints are ECN-capable, so this would be an ECN-PDU. However, ECN-PDU is intended to be a general term for a PDU at lower layers, as well as at the IP layer.

Not-ECN-PDU: A PDU at the IP layer or below that is part of a congestion control feedback-loop within which at least one node necessary to propagate any explicit congestion notification signals back to the Load Regulator is not capable of doing that propagation.

3. Modes of Operation

This section sets down the different modes by which congestion information is passed between the lower layer and the higher one. It acts as a reference framework for the following sections, which give normative guidelines for designers of explicit congestion notification protocols, taking each mode in turn:

Feed-Forward-and-Up: Nodes feed forward congestion notification towards the egress within the lower layer then up and along the layers towards the end-to-end destination at the transport layer. The following local optimisation is possible:

Feed-Up-and-Forward: A lower layer switch feeds-up congestion notification directly into the higher layer (e.g. into the ECN field in the IP header), irrespective of whether the node is at the egress of a subnet.

Feed-Backward: Nodes feed back congestion signals towards the ingress of the lower layer and (optionally) attempt to control congestion within their own layer.

Null: Nodes cannot experience congestion at the lower layer except at ingress nodes (which are IP-aware or equivalently higher-layer-aware).

3.1. Feed-Forward-and-Up Mode

Like IP and MPLS, many subnet technologies are based on self-contained protocol data units (PDUs) or frames sent unreliably. They provide no feedback channel at the subnetwork layer, instead relying on higher layers (e.g. TCP) to feed back loss signals.

In these cases, ECN may best be supported by standardising explicit notification of congestion into the lower layer protocol that carries the data forwards. Then a specification is needed for how the egress of the lower layer subnet propagates this explicit signal into the forwarded upper layer (IP) header. This signal continues forwards until it finally reaches the destination transport (at L4). Then typically the destination will feed this congestion notification back to the source transport using an end-to-end protocol (e.g. TCP). This is the arrangement that has already been used to add ECN to IP-in-IP tunnels [RFC6040], IP-in-MPLS and MPLS-in-MPLS [RFC5129].

This mode is illustrated in Figure 1. Along the middle of the figure, layers 2, 3 and 4 of the protocol stack are shown, and one packet is shown along the bottom as it progresses across the network from source to destination, crossing two subnets connected by a

router, and crossing two switches on the path across each subnet. Congestion at the output of the first switch (shown as *) leads to a congestion marking in the L2 header (shown as C in the illustration of the packet). The chevrons show the progress of the resulting congestion indication. It is propagated from link to link across the subnet in the L2 header, then when the router removes the marked L2 header, it propagates the marking up into the L3 (IP) header. The router forwards the marked L3 header into subnet 2, and when it adds a new L2 header it copies the L3 marking into the L2 header as well, as shown by the 'C's in both layers (assuming the technology of subnet 2 also supports explicit congestion marking).

Note that there is no implication that each 'C' marking is encoded the same; a different encoding might be used for the 'C' marking in each protocol.

Finally, for completeness, we show the L3 marking arriving at the destination, where the host transport protocol (e.g. TCP) feeds it back to the source in the L4 acknowledgement (the 'C' at L4 in the packet at the top of the diagram).

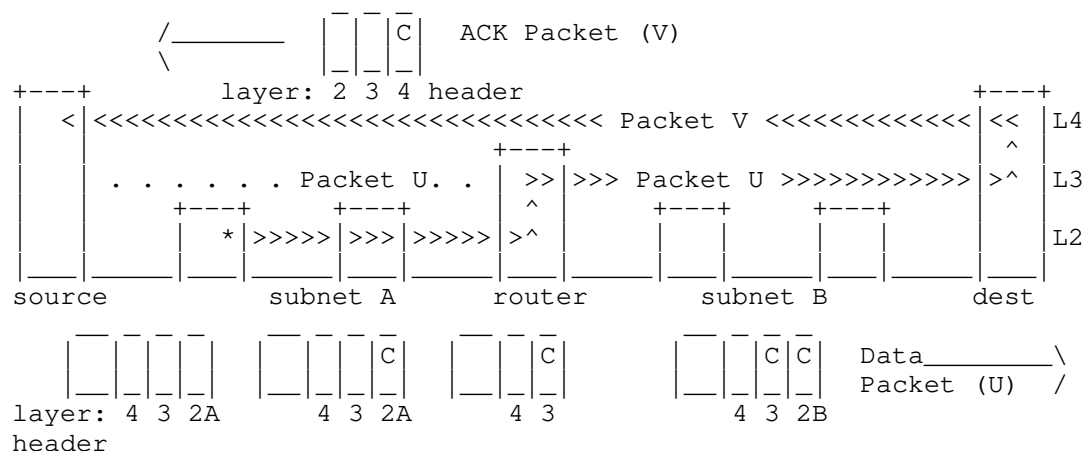


Figure 1: Feed-Forward-and-Up Mode

Of course, modern networks are rarely as simple as this text-book example, often involving multiple nested layers. For example, a 3GPP mobile network may have two IP-in-IP (GTP [GTPv1]) tunnels in series and an MPLS backhaul between the base station and the first router. Nonetheless, the example illustrates the general idea of feeding congestion notification forward then upward whenever a header is removed at the egress of a subnet.

Note that the FECN (forward ECN) bit in Frame Relay [Buck00] and the explicit forward congestion indication (EFCI [ITU-T.I.371]) bit in ATM user data cells follow a feed-forward pattern. However, in ATM, this arrangement is only part of a feed-forward-and-backward pattern at the lower layer, not feed-forward-and-up out of the lower layer--the intention was never to interface to IP ECN at the subnet egress. To our knowledge, Frame Relay FECN is solely used to detect where more capacity should be provisioned.

3.2. Feed-Up-and-Forward Mode

Ethernet is particularly difficult to extend incrementally to support explicit congestion notification. One way to support ECN in such cases has been to use so called 'layer-3 switches'. These are Ethernet switches that dig into the Ethernet payload to find an IP header and manipulate or act on certain IP fields (specifically Diffserv & ECN). For instance, in Data Center TCP [RFC8257], layer-3 switches are configured to mark the ECN field of the IP header within the Ethernet payload when their output buffer becomes congested. With respect to switching, a layer-3 switch acts solely on the addresses in the Ethernet header; it does not use IP addresses, and it does not decrement the TTL field in the IP header.

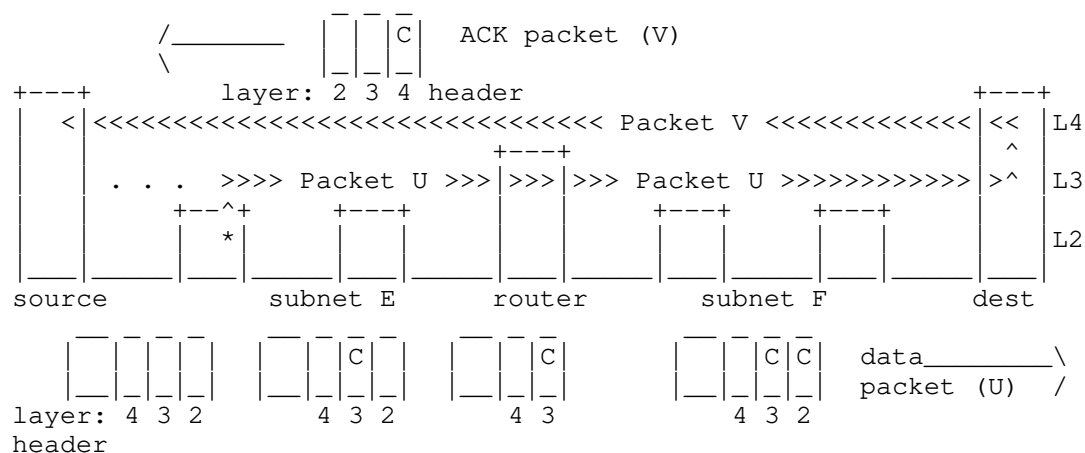


Figure 2: Feed-Up-and-Forward Mode

By comparing Figure 2 with Figure 1, it can be seen that subnet E (perhaps a subnet of layer-3 Ethernet switches) works in feed-up-and-forward mode by notifying congestion directly into L3 at the point of congestion, even though the congested switch does not otherwise act at L3. In this example, the technology in subnet F (e.g. MPLS) does

support ECN natively, so when the router adds the layer-2 header it copies the ECN marking from L3 to L2 as well.

3.3. Feed-Backward Mode

In some layer 2 technologies, explicit congestion notification has been defined for use internally within the subnet with its own feedback and load regulation, but typically the interface with IP for ECN has not been defined.

For instance, for the available bit-rate (ABR) service in ATM, the relative rate mechanism was one of the more popular mechanisms for managing traffic, tending to supersede earlier designs. In this approach ATM switches send special resource management (RM) cells in both the forward and backward directions to control the ingress rate of user data into a virtual circuit. If a switch buffer is approaching congestion or is congested it sends an RM cell back towards the ingress with respectively the No Increase (NI) or Congestion Indication (CI) bit set in its message type field [ATM-TM-ABR]. The ingress then holds or decreases its sending bit-rate accordingly.

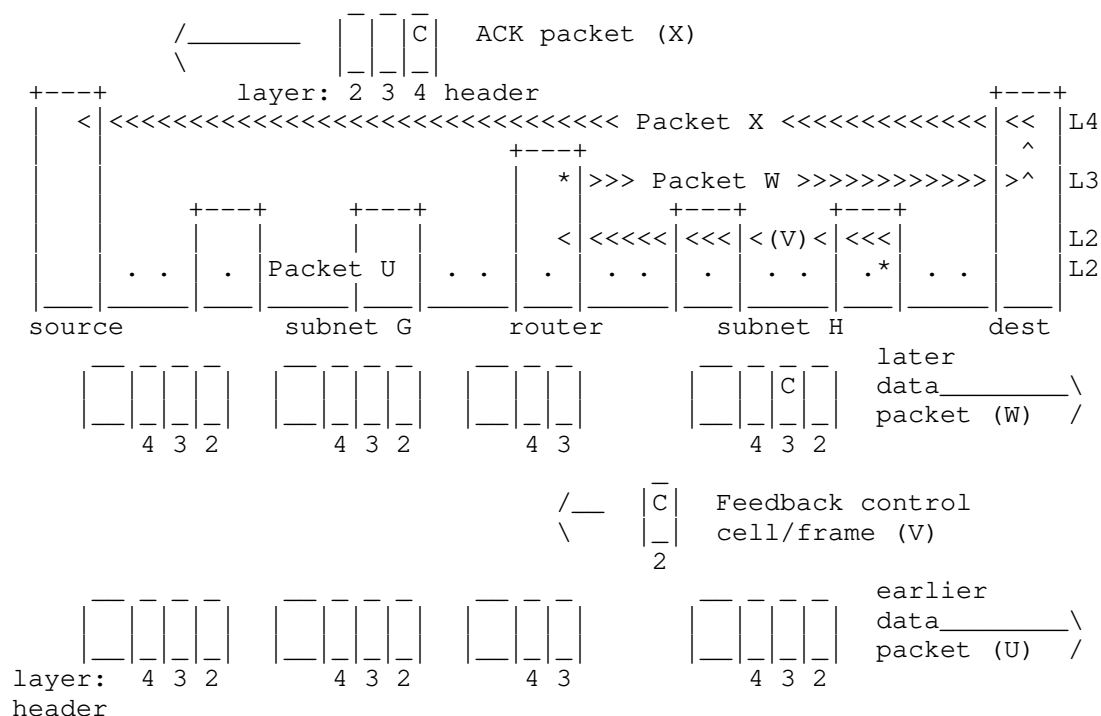


Figure 3: Feed-Backward Mode

ATM's feed-backward approach does not fit well when layered beneath IP's feed-forward approach--unless the initial data source is the same node as the ATM ingress. Figure 3 shows the feed-backward approach being used in subnet H. If the final switch on the path is congested (*), it does not feed-forward any congestion indications on packet (U). Instead it sends a control cell (V) back to the router at the ATM ingress.

However, the backward feedback does not reach the original data source directly because IP does not support backward feedback (and subnet G is independent of subnet H). Instead, the router in the middle throttles down its sending rate but the original data sources don't reduce their rates. The resulting rate mismatch causes the middle router's buffer at layer 3 to back up until it becomes congested, which it signals forwards on later data packets at layer 3 (e.g. packet W). Note that the forward signal from the middle router is not triggered directly by the backward signal. Rather, it is triggered by congestion resulting from the middle router's mismatched rate response to the backward signal.

In response to this later forward signalling, end-to-end feedback at layer-4 finally completes the tortuous path of congestion indications back to the origin data source, as before.

Quantized congestion notification (QCN [IEEE802.1Q]) would suffer from similar problems if extended to multiple subnets. However, from the start QCN was clearly characterized as solely applicable to a single subnet (see Section 6).

3.4. Null Mode

Often link and physical layer resources are 'non-blocking' by design. In these cases congestion notification may be implemented but it does not need to be deployed at the lower layer; ECN in IP would be sufficient.

A degenerate example is a point-to-point Ethernet link. Excess loading of the link merely causes the queue from the higher layer to back up, while the lower layer remains immune to congestion. Even a whole meshed subnetwork can be made immune to interior congestion by limiting ingress capacity and sufficient sizing of interior links, e.g. a non-blocking fat-tree network [Leiserson85]. An alternative to fat links near the root is numerous thin links with multi-path routing to ensure even worst-case patterns of load cannot congest any link, e.g. a Clos network [Clos53].

4. Feed-Forward-and-Up Mode: Guidelines for Adding Congestion Notification

Feed-forward-and-up is the mode already used for signalling ECN up the layers through MPLS into IP [RFC5129] and through IP-in-IP tunnels [RFC6040], whether encapsulating with IPv4 [RFC2003], IPv6 [RFC2473] or IPsec [RFC4301]. These RFCs take a consistent approach and the following guidelines are designed to ensure this consistency continues as ECN support is added to other protocols that encapsulate IP. The guidelines are also designed to ensure compliance with the more general best current practice for the design of alternate ECN schemes given in [RFC4774] and extended by [RFC8311].

The rest of this section is structured as follows:

- o Section 4.1 addresses the most straightforward cases, where [RFC6040] can be applied directly to add ECN to tunnels that are effectively IP-in-IP tunnels, but with shim header(s) between the IP headers.
- o The subsequent sections give guidelines for adding ECN to a subnet technology that uses feed-forward-and-up mode like IP, but it is

not so similar to IP that [RFC6040] rules can be applied directly. Specifically:

- * Sections 4.2, 4.3 and 4.4 respectively address how to add ECN support to the wire protocol and to the encapsulators and decapsulators at the ingress and egress of the subnet.
- * Section 4.5 deals with the special, but common, case of sequences of tunnels or subnets that all use the same technology
- * Section 4.6 deals with the question of reframing when IP packets do not map 1:1 into lower layer frames.

4.1. IP-in-IP Tunnels with Shim Headers

A common pattern for many tunnelling protocols is to encapsulate an inner IP header with shim header(s) then an outer IP header. A shim header is defined as one that is not sufficient alone to forward the packet as an outer header. Another common pattern is for a shim to encapsulate a layer 2 (L2) header, which in turn encapsulates (or might encapsulate) an IP header. [I-D.ietf-tsvwg-rfc6040update-shim] clarifies that RFC 6040 is just as applicable when there are shim(s) and possibly a L2 header between two IP headers.

However, it is not always feasible or necessary to propagate ECN between IP headers when separated by a shim. For instance, it might be too costly to dig to arbitrary depths to find an inner IP header, there may be little or no congestion within the tunnel by design (see null mode in Section 3.4 above), or a legacy implementation might not support ECN. In cases where a tunnel does not support ECN, it is important that the ingress does not copy the ECN field from an inner IP header to an outer. Therefore section 4 of [I-D.ietf-tsvwg-rfc6040update-shim] requires network operators to configure the ingress of a tunnel that does not support ECN so that it zeros the ECN field in the outer IP header.

Nonetheless, in many cases it is feasible to propagate the ECN field between IP headers separated by shim header(s) and/or a L2 header. Particularly in the typical case when the outer IP header and the shim(s) are added (or removed) as part of the same procedure. Even if the shim(s) encapsulate a L2 header, it is often possible to find an inner IP header within the L2 PDU and propagate ECN between that and the outer IP header. This can be thought of as a special case of the feed-up-and-forward mode (Section 3.2), so the guidelines for this mode apply (Section 5).

Numerous shim protocols have been defined for IP tunnelling. More recent ones e.g. Geneve [RFC8926] and Generic UDP Encapsulation (GUE) [I-D.ietf-intarea-gue] cite and follow RFC 6040. And some earlier ones, e.g. CAPWAP [RFC5415] and LISP [RFC6830], cite RFC 3168, which is compatible with RFC 6040.

However, as Section 9.3 of RFC 3168 pointed out, ECN support needs to be defined for many earlier shim-based tunnelling protocols, e.g. L2TPv2 [RFC2661], L2TPv3 [RFC3931], GRE [RFC2784], PPTP [RFC2637], GTP [GTPv1], [GTPv1-U], [GTPv2-C] and Teredo [RFC4380] as well as some recent ones, e.g. VXLAN [RFC7348], NVGRE [RFC7637] and NSH [RFC8300].

All these IP-based encapsulations can be updated in one shot by simple reference to RFC 6040. However, it would not be appropriate to update all these protocols from within the present guidance document. Instead a companion specification [I-D.ietf-tsvwg-rfc6040update-shim] has been prepared that has the appropriate standards track status to update standards track protocols. For those that are not under IETF change control [I-D.ietf-tsvwg-rfc6040update-shim] can only recommend that the relevant body updates them.

4.2. Wire Protocol Design: Indication of ECN Support

This section is intended to guide the redesign of any lower layer protocol that encapsulate IP to add native ECN support at the lower layer. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

A lower layer (or subnet) congestion notification system:

1. SHOULD NOT apply explicit congestion notifications to PDUs that are destined for legacy layer-4 transport implementations that will not understand ECN, and
2. SHOULD NOT apply explicit congestion notifications to PDUs if the egress of the subnet might not propagate congestion notifications onward into the higher layer.

We use the term ECN-PDUs for a PDU on a feedback loop that will propagate congestion notification properly because it meets both the above criteria. And a Not-ECN-PDU is a PDU on a feedback loop that does not meet at least one of the criteria, and will therefore not propagate congestion notification properly. A

corollary of the above is that a lower layer congestion notification protocol:

3. SHOULD be able to distinguish ECN-PDUs from Not-ECN-PDUs.

Note that there is no need for all interior nodes within a subnet to be able to mark congestion explicitly. A mix of ECN and drop signals from different nodes is fine. However, if any interior nodes might generate ECN markings, guideline 2 above says that all relevant egress node(s) SHOULD be able to propagate those markings up to the higher layer.

In IP, if the ECN field in each PDU is cleared to the Not-ECT (not ECN-capable transport) codepoint, it indicates that the L4 transport will not understand congestion markings. A congested buffer must not mark these Not-ECT PDUs, and therefore drops them instead.

The mechanism a lower layer uses to distinguish the ECN-capability of PDUs need not mimic that of IP. The above guidelines merely say that the lower layer system, as a whole, should achieve the same outcome. For instance, ECN-capable feedback loops might use PDUs that are identified by a particular set of labels or tags. Alternatively, logical link protocols that use flow state might determine whether a PDU can be congestion marked by checking for ECN-support in the flow state. Other protocols might depend on out-of-band control signals.

The per-domain checking of ECN support in MPLS [RFC5129] is a good example of a way to avoid sending congestion markings to L4 transports that will not understand them, without using any header space in the subnet protocol.

In MPLS, header space is extremely limited, therefore RFC5129 does not provide a field in the MPLS header to indicate whether the PDU is an ECN-PDU or a Not-ECN-PDU. Instead, interior nodes in a domain are allowed to set explicit congestion indications without checking whether the PDU is destined for a L4 transport that will understand them. Nonetheless, this is made safe by requiring that the network operator upgrades all decapsulating edges of a whole domain at once, as soon as even one switch within the domain is configured to mark rather than drop during congestion. Therefore, any edge node that might decapsulate a packet will be capable of checking whether the higher layer transport is ECN-capable. When decapsulating a CE-marked packet, if the decapsulator discovers that the higher layer (inner header) indicates the transport is not ECN-capable, it drops the packet--effectively on behalf of the earlier congested node (see Decapsulation Guideline 1 in Section 4.4).

It was only appropriate to define such an incremental deployment strategy because MPLS is targeted solely at professional operators, who can be expected to ensure that a whole subnetwork is consistently configured. This strategy might not be appropriate for other link technologies targeted at zero-configuration deployment or deployment by the general public (e.g. Ethernet). For such 'plug-and-play' environments it will be necessary to invent a failsafe approach that ensures congestion markings will never fall into black holes, no matter how inconsistently a system is put together. Alternatively, congestion notification relying on correct system configuration could be confined to flavours of Ethernet intended only for professional network operators, such as Provider Backbone Bridges (PBB [IEEE802.1Q]; previously 802.1ah).

ECN support in TRILL [I-D.ietf-trill-ecn-support] provides a good example of how to add ECN to a lower layer protocol without relying on careful and consistent operator configuration. TRILL provides an extension header word with space for flags of different categories depending on whether logic to understand the extension is critical. The congestion experienced marking has been defined as a 'critical ingress-to-egress' flag. So if a transit RBridge sets this flag and an egress RBridge does not have any logic to process it, it will drop it; which is the desired default action anyway. Therefore TRILL RBridges can be updated with support for ECN in no particular order and, at the egress of the TRILL campus, congestion notification will be propagated to IP as ECN whenever ECN logic has been implemented, or as drop otherwise.

QCN [IEEE802.1Q] is not intended to extend beyond a single subnet, or to interoperate with ECN. Nonetheless, the way QCN indicates to lower layer devices that the end-points will not understand QCN provides another example that a lower layer protocol designer might be able to mimic for their scenario. An operator can define certain Priority Code Points (PCPs [IEEE802.1Q]; previously 802.1p) to indicate non-QCN frames and an ingress bridge is required to map arriving not-QCN-capable IP packets to one of these non-QCN PCPs.

4.3. Encapsulation Guidelines

This section is intended to guide the redesign of any node that encapsulates IP with a lower layer header when adding native ECN support to the lower layer protocol. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

1. Egress Capability Check: A subnet ingress needs to be sure that the corresponding egress of a subnet will propagate any

congestion notification added to the outer header across the subnet. This is necessary in addition to checking that an incoming PDU indicates an ECN-capable (L4) transport. Examples of how this guarantee might be provided include:

- * by configuration (e.g. if any label switches in a domain support ECN marking, [RFC5129] requires all egress nodes to have been configured to propagate ECN)
 - * by the ingress explicitly checking that the egress propagates ECN (e.g. an early attempt to add ECN support to TRILL used IS-IS to check path capabilities before adding ECN extension flags to each frame [RFC7780]).
 - * by inherent design of the protocol (e.g. by encoding ECN marking on the outer header in such a way that a legacy egress that does not understand ECN will consider the PDU corrupt or invalid and discard it, thus at least propagating a form of congestion signal).
2. Egress Fails Capability Check: If the ingress cannot guarantee that the egress will propagate congestion notification, the ingress SHOULD disable ECN at the lower layer when it forwards the PDU. An example of how the ingress might disable ECN at the lower layer would be by setting the outer header of the PDU to identify it as a Not-ECN-PDU, assuming the subnet technology supports such a concept.
 3. Standard Congestion Monitoring Baseline: Once the ingress to a subnet has established that the egress will correctly propagate ECN, on encapsulation it SHOULD encode the same level of congestion in outer headers as is arriving in incoming headers. For example it might copy any incoming congestion notification into the outer header of the lower layer protocol.

This ensures that bulk congestion monitoring of outer headers (e.g. by a network management node monitoring ECN in passing frames) will measure congestion accumulated along the whole upstream path - since the Load Regulator not just since the ingress of the subnet. A node that is not the Load Regulator SHOULD NOT re-initialize the level of CE markings in the outer to zero.

It would still also be possible to measure congestion introduced across one subnet (or tunnel) by subtracting the level of CE markings on inner headers from that on outer headers (see Appendix C of [RFC6040]). For example:

- * If this guideline has been followed and if the level of CE markings is 0.4% on the outer and 0.1% on the inner, 0.4% congestion has been introduced across all the networks since the load regulator, and 0.3% ($= 0.4\% - 0.1\%$) has been introduced since the ingress to the current subnet (or tunnel);
- * Without this guideline, if the subnet ingress had re-initialized the outer congestion level to zero, the outer and inner would measure 0.1% and 0.3%. It would still be possible to infer that the congestion introduced since the Load Regulator was 0.4% ($= 0.1\% + 0.3\%$). But only if the monitoring system somehow knows whether the subnet ingress re-initialized the congestion level.

As long as subnet and tunnel technologies use the standard congestion monitoring baseline in this guideline, monitoring systems will know to use the former approach, rather than having to "somehow know" which approach to use.

4.4. Decapsulation Guidelines

This section is intended to guide the redesign of any node that decapsulates IP from within a lower layer header when adding native ECN support to the lower layer protocol. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

A subnet egress SHOULD NOT simply copy congestion notification from outer headers to the forwarded header. It SHOULD calculate the outgoing congestion notification field from the inner and outer headers using the following guidelines. If there is any conflict, rules earlier in the list take precedence over rules later in the list:

1. If the arriving inner header is a Not-ECN-PDU it implies the L4 transport will not understand explicit congestion markings.
Then:
 - * If the outer header carries an explicit congestion marking, drop is the only indication of congestion that the L4 transport will understand. If the congestion marking is the most severe possible, the packet MUST be dropped. However, if congestion can be marked with multiple levels of severity and the packet's marking is not the most severe, this requirement can be relaxed to: the packet SHOULD be dropped.

- * If the outer is an ECN-PDU that carries no indication of congestion or a Not-ECN-PDU the PDU SHOULD be forwarded, but still as a Not-ECN-PDU.
- 2. If the outer header does not support explicit congestion notification (a Not-ECN-PDU), but the inner header does (an ECN-PDU), the inner header SHOULD be forwarded unchanged.
- 3. In some lower layer protocols congestion may be signalled as a numerical level, such as in the control frames of quantized congestion notification (QCN [IEEE802.1Q]). If such a multi-bit encoding encapsulates an ECN-capable IP data packet, a function will be needed to convert the quantized congestion level into the frequency of congestion markings in outgoing IP packets.
- 4. Congestion indications might be encoded by a severity level. For instance increasing levels of congestion might be encoded by numerically increasing indications, e.g. pre-congestion notification (PCN) can be encoded in each PDU at three severity levels in IP or MPLS [RFC6660] and the default encapsulation and decapsulation rules [RFC6040] are compatible with this interpretation of the ECN field.

If the arriving inner header is an ECN-PDU, where the inner and outer headers carry indications of congestion of different severity, the more severe indication SHOULD be forwarded in preference to the less severe.

- 5. The inner and outer headers might carry a combination of congestion notification fields that should not be possible given any currently used protocol transitions. For instance, if Encapsulation Guideline 3 in Section 4.3 had been followed, it should not be possible to have a less severe indication of congestion in the outer than in the inner. It MAY be appropriate to log unexpected combinations of headers and possibly raise an alarm.

If a safe outgoing codepoint can be defined for such a PDU, the PDU SHOULD be forwarded rather than dropped. Some implementers discard PDUs with currently unused combinations of headers just in case they represent an attack. However, an approach using alarms and policy-mediated drop is preferable to hard-coded drop, so that operators can keep track of possible attacks but currently unused combinations are not precluded from future use through new standards actions.

4.5. Sequences of Similar Tunnels or Subnets

In some deployments, particularly in 3GPP networks, an IP packet may traverse two or more IP-in-IP tunnels in sequence that all use identical technology (e.g. GTP).

In such cases, it would be sufficient for every encapsulation and decapsulation in the chain to comply with RFC 6040. Alternatively, as an optimisation, a node that decapsulates a packet and immediately re-encapsulates it for the next tunnel MAY copy the incoming outer ECN field directly to the outgoing outer and the incoming inner ECN field directly to the outgoing inner. Then the overall behavior across the sequence of tunnel segments would still be consistent with RFC 6040.

Appendix C of RFC6040 describes how a tunnel egress can monitor how much congestion has been introduced within a tunnel. A network operator might want to monitor how much congestion had been introduced within a whole sequence of tunnels. Using the technique in Appendix C of RFC6040 at the final egress, the operator could monitor the whole sequence of tunnels, but only if the above optimisation were used consistently along the sequence of tunnels, in order to make it appear as a single tunnel. Therefore, tunnel endpoint implementations SHOULD allow the operator to configure whether this optimisation is enabled.

When ECN support is added to a subnet technology, consideration SHOULD be given to a similar optimisation between subnets in sequence if they all use the same technology.

4.6. Reframing and Congestion Markings

The guidance in this section is worded in terms of framing boundaries, but it applies equally whether the protocol data units are frames, cells or packets.

Where an AQM marks the ECN field of IP packets as they queue into a layer-2 link, there will be no problem with framing boundaries, because the ECN markings would be applied directly to IP packets. The guidance in this section is only applicable where an ECN capability is being added to a layer-2 protocol so that layer-2 frames can be ECN-marked by an AQM at layer-2. This would only be necessary where AQM will be applied at pure layer-2 nodes (without IP-awareness).

When layer-2 frame headers are stripped off and IP PDUs with different boundaries are forwarded, the provisions in RFC7141 for handling congestion indications when splitting or merging packets

apply (see Section 2.4 of [RFC7141]). Those provisions include: "The general rule to follow is that the number of octets in packets with congestion indications SHOULD be equivalent before and after merging or splitting." See RFC 7141 for the complete provisions and related discussion, including an exception to that general rule.

As also recommended in RFC 7141, the mechanism for propagating congestion indications SHOULD ensure that any new incoming congestion indication is propagated immediately, and not held awaiting possible arrival of further congestion indications sufficient to indicate congestion for all of the octets of an outgoing IP PDU.

5. Feed-Up-and-Forward Mode: Guidelines for Adding Congestion Notification

The guidance in this section is applicable, for example, when IP packets:

- o are encapsulated in Ethernet headers, which have no support for ECN;
- o are forwarded by the eNode-B (base station) of a 3GPP radio access network, which is required to apply ECN marking during congestion, [LTE-RA], [UTRAN], but the Packet Data Convergence Protocol (PDCP) that encapsulates the IP header over the radio access has no support for ECN.

This guidance also generalizes to encapsulation by other subnet technologies with no native support for explicit congestion notification at the lower layer, but with support for finding and processing an IP header. It is unlikely to be applicable or necessary for IP-in-IP encapsulation, where feed-forward-and-up mode based on [RFC6040] would be more appropriate.

Marking the IP header while switching at layer-2 (by using a layer-3 switch) or while forwarding in a radio access network seems to represent a layering violation. However, it can be considered as a benign optimisation if the guidelines below are followed. Feed-up-and-forward is certainly not a general alternative to implementing feed-forward congestion notification in the lower layer, because:

- o IPv4 and IPv6 are not the only layer-3 protocols that might be encapsulated by lower layer protocols
- o Link-layer encryption might be in use, making the layer-2 payload inaccessible

- o Many Ethernet switches do not have 'layer-3 switch' capabilities so they cannot read or modify an IP payload
- o It might be costly to find an IP header (v4 or v6) when it may be encapsulated by more than one lower layer header, e.g. Ethernet MAC in MAC ([IEEE802.1Q]; previously 802.1ah).

Nonetheless, configuring lower layer equipment to look for an ECN field in an encapsulated IP header is a useful optimisation. If the implementation follows the guidelines below, this optimisation does not have to be confined to a controlled environment such as within a data centre; it could usefully be applied on any network--even if the operator is not sure whether the above issues will never apply:

1. If a native lower-layer congestion notification mechanism exists for a subnet technology, it is safe to mix feed-up-and-forward with feed-forward-and-up on other switches in the same subnet. However, it will generally be more efficient to use the native mechanism.
 2. The depth of the search for an IP header SHOULD be limited. If an IP header is not found soon enough, or an unrecognized or unreadable header is encountered, the switch SHOULD resort to an alternative means of signalling congestion (e.g. drop, or the native lower layer mechanism if available).
 3. It is sufficient to use the first IP header found in the stack; the egress of the relevant tunnel can propagate congestion notification upwards to any more deeply encapsulated IP headers later.
6. Feed-Backward Mode: Guidelines for Adding Congestion Notification

It can be seen from Section 3.3 that congestion notification in a subnet using feed-backward mode has generally not been designed to be directly coupled with IP layer congestion notification. The subnet attempts to minimize congestion internally, and if the incoming load at the ingress exceeds the capacity somewhere through the subnet, the layer 3 buffer into the ingress backs up. Thus, a feed-backward mode subnet is in some sense similar to a null mode subnet, in that there is no need for any direct interaction between the subnet and higher layer congestion notification. Therefore no detailed protocol design guidelines are appropriate. Nonetheless, a more general guideline is appropriate:

A subnetwork technology intended to eventually interface to IP SHOULD NOT be designed using only the feed-backward mode, which is certainly best for a stand-alone subnet, but would need to be

modified to work efficiently as part of the wider Internet, because IP uses feed-forward-and-up mode.

The feed-backward approach at least works beneath IP, where the term 'works' is used only in a narrow functional sense because feed-backward can result in very inefficient and sluggish congestion control--except if it is confined to the subnet directly connected to the original data source, when it is faster than feed-forward. It would be valid to design a protocol that could work in feed-backward mode for paths that only cross one subnet, and in feed-forward-and-up mode for paths that cross subnets.

In the early days of TCP/IP, a similar feed-backward approach was tried for explicit congestion signalling, using source-quench (SQ) ICMP control packets. However, SQ fell out of favour and is now formally deprecated [RFC6633]. The main problem was that it is hard for a data source to tell the difference between a spoofed SQ message and a quench request from a genuine buffer on the path. It is also hard for a lower layer buffer to address an SQ message to the original source port number, which may be buried within many layers of headers, and possibly encrypted.

QCN (also known as backward congestion notification, BCN; see Sections 30--33 of [IEEE802.1Q]; previously known as 802.1Qau) uses a feed-backward mode structurally similar to ATM's relative rate mechanism. However, QCN confines its applicability to scenarios such as some data centres where all endpoints are directly attached by the same Ethernet technology. If a QCN subnet were later connected into a wider IP-based internetwork (e.g. when attempting to interconnect multiple data centres) it would suffer the inefficiency shown in Figure 3.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

If a lower layer wire protocol is redesigned to include explicit congestion signalling in-band in the protocol header, care SHOULD be taken to ensure that the field used is specified as mutable during transit. Otherwise interior nodes signalling congestion would invalidate any authentication protocol applied to the lower layer header--by altering a header field that had been assumed as immutable.

The redesign of protocols that encapsulate IP in order to propagate congestion signals between layers raises potential signal integrity

concerns. Experimental or proposed approaches exist for assuring the end-to-end integrity of in-band congestion signals, e.g.:

- o Congestion exposure (ConEx) for networks to audit that their congestion signals are not being suppressed by other networks or by receivers, and for networks to police that senders are responding sufficiently to the signals, irrespective of the L4 transport protocol used [RFC7713].
- o A test for a sender to detect whether a network or the receiver is suppressing congestion signals (for example see 2nd para of Section 20.2 of [RFC3168]).

Given these end-to-end approaches are already being specified, it would make little sense to attempt to design hop-by-hop congestion signal integrity into a new lower layer protocol, because end-to-end integrity inherently achieves hop-by-hop integrity.

Section 6 gives vulnerability to spoofing as one of the reasons for deprecating feed-backward mode.

9. Conclusions

Following the guidance in this document enables ECN support to be extended to numerous protocols that encapsulate IP (v4 & v6) in a consistent way, so that IP continues to fulfil its role as an end-to-end interoperability layer. This includes:

- o A wide range of tunnelling protocols including those with various forms of shim header between two IP headers, possibly also separated by a L2 header;
- o A wide range of subnet technologies, particularly those that work in the same 'feed-forward-and-up' mode that is used to support ECN in IP and MPLS.

Guidelines have been defined for supporting propagation of ECN between Ethernet and IP on so-called Layer-3 Ethernet switches, using a 'feed-up-and-forward' mode. This approach could enable other subnet technologies to pass ECN signals into the IP layer, even if they do not support ECN natively.

Finally, attempting to add ECN to a subnet technology in feed-backward mode is deprecated except in special cases, due to its likely sluggish response to congestion.

10. Acknowledgements

Thanks to Gorry Fairhurst and David Black for extensive reviews. Thanks also to the following reviewers: Joe Touch, Andrew McGregor, Richard Scheffenegger, Ingemar Johansson, Piers O'Hanlon, Donald Eastlake, Jonathan Morton and Michael Welzl, who pointed out that lower layer congestion notification signals may have different semantics to those in IP. Thanks are also due to the tsvwg chairs, TSV ADs and IETF liaison people such as Eric Gray, Dan Romascanu and Gonzalo Camarillo for helping with the liaisons with the IEEE and 3GPP. And thanks to Georg Mayer and particularly to Erik Guttman for the extensive search and categorisation of any 3GPP specifications that cite ECN specifications.

Bob Briscoe was part-funded by the European Community under its Seventh Framework Programme through the Trilogy project (ICT-216372) for initial drafts and through the Reducing Internet Transport Latency (RITE) project (ICT-317700) subsequently. The views expressed here are solely those of the authors.

11. Contributors

Pat Thaler
Broadcom Corporation (retired)
CA
USA

Pat was a co-author of this draft, but retired before its publication.

12. 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.

13. References

13.1. Normative References

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

- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <<https://www.rfc-editor.org/info/rfc3819>>.
- [RFC4774] Floyd, S., "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field", BCP 124, RFC 4774, DOI 10.17487/RFC4774, November 2006, <<https://www.rfc-editor.org/info/rfc4774>>.
- [RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <<https://www.rfc-editor.org/info/rfc5129>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.
- [RFC7141] Briscoe, B. and J. Manner, "Byte and Packet Congestion Notification", BCP 41, RFC 7141, DOI 10.17487/RFC7141, February 2014, <<https://www.rfc-editor.org/info/rfc7141>>.

13.2. Informative References

- [ATM-TM-ABR] Cisco, "Understanding the Available Bit Rate (ABR) Service Category for ATM VCs", Design Technote 10415, June 2005.
- [Buck00] Buckwalter, J., "Frame Relay: Technology and Practice", Pub. Addison Wesley ISBN-13: 978-0201485240, 2000.
- [Clos53] Clos, C., "A Study of Non-Blocking Switching Networks", Bell Systems Technical Journal 32(2):406--424, March 1953.
- [GTPv1] 3GPP, "GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface", Technical Specification TS 29.060.
- [GTPv1-U] 3GPP, "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", Technical Specification TS 29.281.

- [GTPv2-C] 3GPP, "Evolved General Packet Radio Service (GPRS) Tunneling Protocol for Control plane (GTPv2-C)", Technical Specification TS 29.274.
- [I-D.ietf-intarea-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-09 (work in progress), October 2019.
- [I-D.ietf-trill-ecn-support]
Eastlake, D. E. and B. Briscoe, "TRILL (Transparent Interconnection of Lots of Links): ECN (Explicit Congestion Notification) Support", draft-ietf-trill-ecn-support-07 (work in progress), February 2018.
- [I-D.ietf-tsvwg-ecn-l4s-id]
Schepper, K. D. and B. Briscoe, "Explicit Congestion Notification (ECN) Protocol for Ultra-Low Queuing Delay (L4S)", draft-ietf-tsvwg-ecn-l4s-id-14 (work in progress), March 2021.
- [I-D.ietf-tsvwg-rfc6040update-shim]
Briscoe, B., "Propagating Explicit Congestion Notification Across IP Tunnel Headers Separated by a Shim", draft-ietf-tsvwg-rfc6040update-shim-13 (work in progress), March 2021.
- [IEEE802.1Q]
IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Virtual Bridged Local Area Networks--Amendment 6: Provider Backbone Bridges", IEEE Std 802.1Q-2018, July 2018, <<https://ieeexplore.ieee.org/document/8403927>>.
- [ITU-T.I.371]
ITU-T, "Traffic Control and Congestion Control in B-ISDN", ITU-T Rec. I.371 (03/04), March 2004, <<http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5454061>>.
- [Leiserson85]
Leiserson, C., "Fat-trees: universal networks for hardware-efficient supercomputing", IEEE Transactions on Computers 34(10):892-901, October 1985.
- [LTE-RA] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2", Technical Specification TS 36.300.

- [RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <<https://www.rfc-editor.org/info/rfc2003>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, DOI 10.17487/RFC2637, July 1999, <<https://www.rfc-editor.org/info/rfc2637>>.
- [RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, DOI 10.17487/RFC2661, August 1999, <<https://www.rfc-editor.org/info/rfc2661>>.
- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <<https://www.rfc-editor.org/info/rfc2784>>.
- [RFC2884] Hadi Salim, J. and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", RFC 2884, DOI 10.17487/RFC2884, July 2000, <<https://www.rfc-editor.org/info/rfc2884>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<https://www.rfc-editor.org/info/rfc2983>>.
- [RFC3931] Lau, J., Ed., Townsley, M., Ed., and I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, DOI 10.17487/RFC3931, March 2005, <<https://www.rfc-editor.org/info/rfc3931>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <<https://www.rfc-editor.org/info/rfc4380>>.

- [RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <<https://www.rfc-editor.org/info/rfc5415>>.
- [RFC6633] Gont, F., "Deprecation of ICMP Source Quench Messages", RFC 6633, DOI 10.17487/RFC6633, May 2012, <<https://www.rfc-editor.org/info/rfc6633>>.
- [RFC6660] Briscoe, B., Moncaster, T., and M. Menth, "Encoding Three Pre-Congestion Notification (PCN) States in the IP Header Using a Single Diffserv Codepoint (DSCP)", RFC 6660, DOI 10.17487/RFC6660, July 2012, <<https://www.rfc-editor.org/info/rfc6660>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <<https://www.rfc-editor.org/info/rfc7323>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <<https://www.rfc-editor.org/info/rfc7567>>.
- [RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <<https://www.rfc-editor.org/info/rfc7637>>.
- [RFC7713] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements", RFC 7713, DOI 10.17487/RFC7713, December 2015, <<https://www.rfc-editor.org/info/rfc7713>>.

- [RFC7780] Eastlake 3rd, D., Zhang, M., Perlman, R., Banerjee, A., Ghanwani, A., and S. Gupta, "Transparent Interconnection of Lots of Links (TRILL): Clarifications, Corrections, and Updates", RFC 7780, DOI 10.17487/RFC7780, February 2016, <<https://www.rfc-editor.org/info/rfc7780>>.
- [RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <<https://www.rfc-editor.org/info/rfc8084>>.
- [RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <<https://www.rfc-editor.org/info/rfc8087>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8257] Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L., and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers", RFC 8257, DOI 10.17487/RFC8257, October 2017, <<https://www.rfc-editor.org/info/rfc8257>>.
- [RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <<https://www.rfc-editor.org/info/rfc8300>>.
- [RFC8311] Black, D., "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation", RFC 8311, DOI 10.17487/RFC8311, January 2018, <<https://www.rfc-editor.org/info/rfc8311>>.
- [RFC8926] Gross, J., Ed., Ganga, I., Ed., and T. Sridhar, Ed., "Geneve: Generic Network Virtualization Encapsulation", RFC 8926, DOI 10.17487/RFC8926, November 2020, <<https://www.rfc-editor.org/info/rfc8926>>.
- [UTRAN] 3GPP, "UTRAN Overall Description", Technical Specification TS 25.401.

Appendix A. Changes in This Version (to be removed by RFC Editor)

From ietf-12 to ietf-13

* Following 3rd tsvwg WGLC:

- + Formalized update to RFC 3819 in its own subsection (1.1) and referred to it in the abstract
- + Scope: Clarified that the specification of alternative ECN semantics using ECT(1) was not in RFC 4774, but rather in RFC 8311, and that the problem with using a DSCP to indicate alternative semantics has issues at domain boundaries as well as tunnels.
- + Terminology: tightened up definitions of ECN-PDU and Not-ECN-PDU, and removed definition of Congestion Baseline, given it was only used once.
- + Mentioned QCN where feed-backward is first introduced (S.3), referring forward to where it is discussed more deeply (S.4).
- + Clarified that IS-IS solution to adding ECN support to TRILL was not pursued
- + Completely rewrote the rationale for the guideline about a Standard Congestion Monitoring Baseline, to focus on standardization of the otherwise unknown scenario used, rather than the relative usefulness of the info in each approach
- + Explained the re-framing problem better and added fragmentation as another possible cause of the problem
- + Acknowledged new reviewers
- + Updated references, replaced citations of 802.1Qau and 802.1ah with rolled up 802.1Q, and added citations of Fat trees and Clos Networks
- + Numerous other editorial improvements

From ietf-11 to ietf-12

* Updated references

From ietf-10 to ietf-11

- * Removed short section (was 3) 'Guidelines for All Cases' because it was out of scope, being covered by RFC 4774. Expanded the Scope section (1.2) to explain all this. Explained that the default encap/decap rules already support certain alternative semantics, particularly all three of the alternative semantics for ECT(1): equivalent to ECT(0) , higher severity than ECT(0), and unmarked but implying different marking semantics from ECT(0).
- * Clarified why the QCN example was being given even though not about increment deployment of ECN
- * Pointed to the spoofing issue with feed-backward mode from the Security Considerations section, to aid security review.
- * Removed any ambiguity in the word 'transport' throughout

From ietf-09 to ietf-10

- * Updated section 5.1 on "IP-in-IP tunnels with Shim Headers" to be consistent with updates to draft-ietf-tsvwg-rfc6040update-shim.
- * Removed reference to the ECN nonce, which has been made historic by RFC 8311
- * Removed "Open Issues" Appendix, given all have been addressed.

From ietf-08 to ietf-09

- * Updated para in Intro that listed all the IP-in-IP tunnelling protocols, to instead refer to draft-ietf-tsvwg-rfc6040update-shim
- * Updated section 5.1 on "IP-in-IP tunnels with Shim Headers" to summarize guidance that has evolved as rfc6040update-shim has developed.

From ietf-07 to ietf-08: Refreshed to avoid expiry. Updated references.

From ietf-06 to ietf-07:

- * Added the people involved in liaisons to the acknowledgements.

From ietf-05 to ietf-06:

- * Introduction: Added GUE and Geneve as examples of tightly coupled shims between IP headers that cite RFC 6040. And added VXLAN to list of those that do not.
- * Replaced normative text about tightly coupled shims between IP headers, with reference to new draft-ietf-tsvwg-rfc6040update-shim
- * Wire Protocol Design: Indication of ECN Support: Added TRILL as an example of a well-design protocol that does not need an indication of ECN support in the wire protocol.
- * Encapsulation Guidelines: In the case of a Not-ECN-PDU with a CE outer, replaced SHOULD be dropped, with explanations of when SHOULD or MUST are appropriate.
- * Feed-Up-and-Forward Mode: Explained examples more carefully, referred to PDCP and cited UTRAN spec as well as E-UTRAN.
- * Updated references.
- * Marked open issues as resolved, but did not delete Open Issues Appendix (yet).

From ietf-04 to ietf-05:

- * Explained why tightly coupled shim headers only "SHOULD" comply with RFC 6040, not "MUST".
- * Updated references

From ietf-03 to ietf-04:

- * Addressed Richard Scheffenegger's review comments: primarily editorial corrections, and addition of examples for clarity.

From ietf-02 to ietf-03:

- * Updated references, ad cited RFC4774.

From ietf-01 to ietf-02:

- * Added Section for guidelines that are applicable in all cases.
- * Updated references.

From ietf-00 to ietf-01: Updated references.

From briscoe-04 to ietf-00: Changed filename following tsvwg adoption.

From briscoe-03 to 04:

- * Re-arranged the introduction to describe the purpose of the document first before introducing ECN in more depth. And clarified the introduction throughout.
- * Added applicability to 3GPP TS 36.300.

From briscoe-02 to 03:

- * Scope section:
 - + Added dependence on correct propagation of traffic class information
 - + For the feed-backward mode, deemed multicast and anycast out of scope
- * Ensured all guidelines referring to subnet technologies also refer to tunnels and vice versa by adding applicability sentences at the start of sections 4.1, 4.2, 4.3, 4.4, 4.6 and 5.
- * Added Security Considerations on ensuring congestion signal fields are classed as immutable and on using end-to-end congestion signal integrity technologies rather than hop-by-hop.

From briscoe-01 to 02:

- * Added authors: JK & PT
- * Added
 - + Section 4.1 "IP-in-IP Tunnels with Tightly Coupled Shim Headers"
 - + Section 4.5 "Sequences of Similar Tunnels or Subnets"
 - + roadmap at the start of Section 4, given the subsections have become quite fragmented.
 - + Section 9 "Conclusions"

- * Clarified why transports are starting to be able to saturate interior links
- * Under Section 1.1, addressed the question of alternative signal semantics and included multicast & anycast.
- * Under Section 3.1, included a 3GPP example.
- * Section 4.2. "Wire Protocol Design":
 - + Altered guideline 2. to make it clear that it only applies to the immediate subnet egress, not later ones
 - + Added a reminder that it is only necessary to check that ECN propagates at the egress, not whether interior nodes mark ECN
 - + Added example of how QCN uses 802.1p to indicate support for QCN.
- * Added references to Appendix C of RFC6040, about monitoring the amount of congestion signals introduced within a tunnel
- * Appendix A: Added more issues to be addressed, including plan to produce a standards track update to IP-in-IP tunnel protocols.
- * Updated acks and references

From briscoe-00 to 01:

- * Intended status: BCP (was Informational) & updates 3819 added.
- * Briefer Introduction: Introductory para justifying benefits of ECN. Moved all but a brief enumeration of modes of operation to their own new section (from both Intro & Scope). Introduced incr. deployment as most tricky part.
- * Tightened & added to terminology section
- * Structured with Modes of Operation, then Guidelines section for each mode.
- * Tightened up guideline text to remove vagueness / passive voice / ambiguity and highlight main guidelines as numbered items.
- * Added Outstanding Document Issues Appendix

* Updated references

Authors' Addresses

Bob Briscoe
Independent
UK

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

John Kaippallimalil
Futurewei
5700 Tennyson Parkway, Suite 600
Plano, Texas 75024
USA

EMail: kjohn@futurewei.com

Transport Working Group
Internet-Draft
Intended status: Informational
Expires: December 29, 2016

T. Szigeti
F. Baker
Cisco Systems
June 27, 2016

Guidelines for DiffServ to IEEE 802.11 Mapping
draft-ietf-tsvwg-ieee-802-11-00

Abstract

As internet traffic is increasingly sourced-from and destined-to wireless endpoints, it is crucial that Quality of Service be aligned between wired and wireless networks; however, this is not always the case by default. This is due to the fact that two independent standards bodies provide QoS guidance on wired and wireless networks: specifically, the IETF specifies standards and design recommendations for wired IP networks, while a separate and autonomous standards-body, the IEEE, administers the standards for wireless 802.11 networks. The purpose of this document is to propose a set Differentiated Services Code Point (DSCP) to IEEE 802.11 User Priority (UP) mappings to reconcile the marking recommendations offered by these two standards bodies, and, as such, to optimize wired-and-wireless interconnect QoS.

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, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Related work	3
1.2. Interaction with RFC 7561	4
1.3. Applicability Statement	4
1.4. Document Organization	5
1.5. Requirements Language	5
2. Comparison and Default Interoperation of DiffServ and IEEE 802.11	5
2.1. Default DSCP-to-UP Mappings and Conflicts	6
2.2. Default UP-to-DSCP Mappings and Conflicts	7
3. Wireless Device Marking and Mapping Capability Recommendations	8
4. DSCP-to-UP Mapping Recommendations	9
4.1. Network Control Traffic	9
4.1.1. Network Control Protocols	9
4.1.2. Operations Administration Management (OAM)	10
4.2. User Traffic	10
4.2.1. Telephony	11
4.2.2. Signaling	11
4.2.3. Multimedia Conferencing	12
4.2.4. Real-Time Interactive	12
4.2.5. Multimedia-Streaming	13
4.2.6. Broadcast Video	13
4.2.7. Low-Latency Data	13
4.2.8. High-Throughput Data	14
4.2.9. Standard Service Class	14
4.2.10. Low-Priority Data	14
4.3. DSCP-to-UP Mapping Recommendations Summary	15
5. Upstream Mapping Recommendations	17
5.1. Upstream DSCP-to-UP Mapping within the Wireless Client Operating System	17
5.2. UP-to-DSCP Mapping at the Wireless Access Point	17
5.3. DSCP-Trust at the Wireless Access Point	18
6. Appendix: IEEE 802.11 QoS Overview	18
6.1. Distributed Coordination Function (DCF)	19
6.1.1. Slot Time	19

6.1.2. Interframe Spaces	20
6.1.3. Contention Windows	20
6.2. Hybrid Coordination Function (HCF)	21
6.2.1. User Priority (UP)	21
6.2.2. Access Category (AC)	21
6.2.3. Arbitration Inter-Frame Space (AIFS)	22
6.2.4. Access Category Contention Windows (CW)	23
6.3. IEEE 802.11u QoS Map Set	24
7. IANA Considerations	25
8. Security Considerations	25
8.1. Privacy Considerations	25
9. Acknowledgements	25
10. References	25
10.1. Normative References	25
10.2. Informative References	26
Appendix A. Change Log	27
Authors' Addresses	27

1. Introduction

Wireless has become the medium of choice for endpoints connecting to business and private networks. However, the wireless medium defined by IEEE 802.11 [IEEE.802-11.2012] presents several design challenges for ensuring end-to-end quality of service. Some of these challenges relate to the nature of 802.11 RF medium itself, being a half-duplex and shared media, while other challenges relate to the fact that the 802.11 standard is not administered by the standards body that administers the rest of the IP network. While the IEEE has developed tools to enable QoS over wireless networks, little guidance exists on how to optimally interconnect wired IP and wireless 802.11 networks, which is the aim of this document.

1.1. Related work

Several RFCs outline DiffServ QoS recommendations over IP networks, including:

- o [RFC2474] specifies the DiffServ Codepoint Field. This RFC also details Class Selectors, as well as the Default Forwarding (DF) treatment.
- o [RFC2475] defines a DiffServ architecture
- o [RFC3246] specifies the Expedited Forwarding (EF) Per-Hop Behavior (PHB)
- o [RFC2597] details the Assured Forwarding (AF) PHB.

- o [RFC3662] outlines a Lower Effort Per-Domain Behavior (PDB)
- o [RFC4594] presents Configuration Guidelines for DiffServ Service Classes
- o [RFC5127] discusses the Aggregation of Diffserv Service Classes
- o [RFC5865] introduces a DSCP for Capacity Admitted Traffic

This draft draws heavily on [RFC4594], [RFC5127], and [I-D.ietf-tsvwg-diffserv-intercon].

In turn, the relevant standard for wireless QoS is IEEE 802.11, which is being progressively updated; the current version of which (at the time of writing) is IEEE 802.11-2012.

1.2. Interaction with RFC 7561

There is also a recommendation from GSMA, Mapping Quality of Service (QoS) Procedures of Proxy Mobile IPv6 (PMIPv6) and WLAN [RFC7561]. The GSMA specification was developed without reference to the service plan documented in Section 1.1, and conflicts both in the services specified and the code points specified for them. As such, the two plans cannot be normalized. Rather, as discussed in [RFC2474] section 2, the two domains (802.11 and GSMA) are different Differentiated Services Domains separated by a Differentiated Services Boundary. At that boundary, code points from one domain are translated to code points for the other, and maybe to Default (zero) if there is no corresponding service to translate to.

1.3. Applicability Statement

This document is applicable to the use of Differentiated Services that interconnect with IEEE 802.11 wireless LANs (referred to as Wi-Fi, throughout this document, for simplicity). These guidelines are applicable whether the wireless access points (APs) are deployed in an autonomous manner, managed by (centralized or distributed) WLAN controllers or some hybrid deployment option. This is because in all these cases, the wireless access point is the bridge between wired and wireless media.

This document primarily applies to wired IP networks that have wireless access points at their edges, but can also be applied to Wi-Fi backhaul, wireless mesh solutions or any other type of AP-to-AP wireless network that serves to extend the IP network infrastructure.

1.4. Document Organization

This document is organized as follows:

- o Section 1 outlines the abstract, related work, organization and the requirements language of this document.
- o Section 2 begins the discussion with a comparison of IETF DiffServ QoS and Wi-Fi QoS standards and highlights discrepancies between these that require reconciliation.
- o Section 3 presents the marking and mapping capabilities that wireless access points and wireless endpoint devices are recommended to support.
- o Section 4 presents DSCP-to-UP mapping recommendations for each of the [RFC4594] traffic classes, which are primarily applicable in the downstream (wired-to-wireless) direction.
- o Section 5, in turn, considers upstream (wireless-to-wired) QoS options, their respective merits and recommendations.
- o Section 6 (in the form of an Appendix) presents a brief overview of how QoS is achieved over IEEE 802.11 wireless networks, given the shared, half-duplex nature of the wireless medium.
- o Section 7 on notes IANA considerations, security considerations, acknowledgements and references, respectively

1.5. Requirements Language

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

2. Comparison and Default Interoperation of DiffServ and IEEE 802.11

(Section 6 provides a brief overview of IEEE 802.11 QoS.)

The following comparisons between IEEE 802.11 and DiffServ should be noted:

- o 802.11 does not support a [RFC3246] EF PHB service, as it is not possible to guarantee that a given access category will be serviced with strict priority over another (due to the random element within the contention process)

- o 802.11 does not support a [RFC2597] AF PHB service, again because it is not possible to guarantee that a given access category will be serviced with a minimum amount of assured bandwidth (due to the non-deterministic nature of the contention process)
- o 802.11 loosely supports a [RFC2474] Default Forwarding service via the Best Effort Access Category (AC_BE)
- o 802.11 loosely supports a [RFC3662] Lower PDB service via the Background Access Category (AC_BK)

As such, these are high-level considerations that need to be kept in mind when mapping from DiffServ to 802.11 (and vice-versa); however, some additional marking-specific incompatibilities must also be reconciled, as will be discussed next.

2.1. Default DSCP-to-UP Mappings and Conflicts

While no explicit guidance is offered in mapping (6-Bit) Layer 3 DSCP values to (3-Bit) Layer 2 markings (such as IEEE 802.1D, 802.1p or 802.11e), a common practice in the networking industry is to map these by what we will refer to as 'Default DSCP-to-UP Mapping' (for lack of a better term), wherein the 3 Most Significant Bits (MSB) of the DSCP are transcribed to generate the corresponding L2 markings.

Note: There are example mappings in IEEE 802.11 (in the Annex V Tables V-1 and V2), but these mappings are provided as examples (vs. as recommendations). Furthermore, some of these mappings do not align with the intent and recommendations expressed in [RFC4594], as will be discussed in the following section.

However, when this default DSCP-to-UP mapping method is applied to packets marked per [RFC4594] recommendations and destined to 802.11 WLAN clients, it will yield a number of sub-optimal QoS mappings, specifically:

- o Voice (EF-101110) will be mapped to UP 5 (101), and treated in the Video Access Category (AC_VI), rather than the Voice Access Category (AC_VO), for which it is intended
- o Multimedia Streaming (AF3-011xx0) will be mapped to UP3 (011) and treated in the Best Effort Access Category (AC_BE), rather than the Video Access Category (AC_VI), for which it is intended
- o OAM traffic (CS2-010000) will be mapped to UP 2 (010) and treated in the Background Access Category (AC_BK), which is not the intent expressed in [RFC4594] for this traffic class

It should also be noted that while IEEE 802.11 defines an intended use for each access category through the AC naming convention (for example, UP 6 and UP 7 belong to AC_VO, the Voice Access Category), 802.11 does not:

- o define how upper Layer markings (such as DSCP) should map to UPs (and hence to ACs)
- o define how UPs should translate to other medium Layer 2 QoS markings
- o strictly restrict each access category to applications reflected in the AC name

2.2. Default UP-to-DSCP Mappings and Conflicts

In the opposite direction of flow (the upstream direction, that is, from wireless-to-wired), many APs use what we will refer to as 'Default UP-to-DSCP Mapping' (for lack of a better term), wherein DSCP values are derived from UP values by multiplying the UP values by 8 (i.e. shifting the 3 UP bits to the left and adding three additional zeros to generate a DSCP value). This derived DSCP value is then used for QoS treatment between the wireless access point and the nearest classification and marking policy enforcement point (which may be the centralized wireless LAN controller, relatively deep within the network).

It goes without saying that when 6 bits of marking granularity are derived from 3, then information is lost in translation. Servicing differentiation cannot be made for 12 classes of traffic (as recommended in [RFC4594]), but for only 8 (with one of these classes being reserved for future use (i.e. UP 7 which maps to DSCP CS7)).

Such default upstream mapping can also yield several inconsistencies with [RFC4594], including:

- o Mapping UP 6 (Voice) to CS6, which [RFC4594] recommends for Network Control
- o Mapping UP 4 (Multimedia Conferencing and/or Real-Time Interactive) to CS4, thus losing the ability to distinguish between these two distinct traffic classes
- o Mapping UP 3 (Multimedia Streaming and/or Broadcast Video) to CS3, thus losing the ability to distinguish between these two distinct traffic classes

- o Mapping UP 2 (Low-Latency Data and/or OAM) to CS2, thus losing the ability to distinguish between these two distinct traffic classes, and possibly overwhelming the queues provisioned for OAM (which is typically lower in capacity [being network control traffic], as compared to Low-Latency Data queues [being user traffic])
- o Mapping UP 1 (High-Throughput Data and/or Low-Priority Data) to CS1, thus losing the ability to distinguish between these two distinct traffic classes and causing legitimate business-relevant High-Throughput Data to receive a [RFC3662] Lower PDB, for which it is not intended

Thus, the next sections of this draft seek to address these limitations and concerns and reconcile the intents of [RFC4594] and IEEE 802.11. First the downstream (wired-to-wireless) DSCP-to-UP mappings will be aligned and then upstream (wireless-to-wired) models will be addressed.

3. Wireless Device Marking and Mapping Capability Recommendations

This document assumes and RECOMMENDS that all wireless access points (as the bridges between wired-and-wireless networks) support the ability to:

- o mark DSCP, per DiffServ standards
- o mark UP, per the 802.11 standard
- o support fully-configurable mappings between DSCP and UP
- o trust the DSCP markings set by wireless endpoint devices (as discussed in Section 5.3)

This document further assumes and RECOMMENDS that all wireless endpoint devices support the ability to:

- o mark DSCP, per DiffServ standards
- o mark UP, per the 802.11 standard
- o support fully-configurable mappings between DSCP (set by applications in software) and UP (set by the operating system and/or wireless network interface hardware drivers)

Having made the assumptions and recommendations above, it bears mentioning while the mappings presented in this document are RECOMMENDED to replace the current common default practices (as discussed in Section 2.1 and Section 2.2), these mapping

recommendations are not expected to fit every last deployment model, and as such may be overridden by network administrators, as needed.

4. DSCP-to-UP Mapping Recommendations

The following section proposes downstream (wired-to-wireless) mappings between [RFC4594] Configuration Guidelines for DiffServ Service Classes and IEEE 802.11. As such, this section draws heavily from [RFC4594], including traffic class definitions and recommendations.

This section assumes wireless access points and/or WLAN controllers that support customizable, non-default DSCP-to-UP mapping schemes.

4.1. Network Control Traffic

Network control traffic is defined as packet flows that are essential for stable operation of the administered network. Network control traffic is different from user application control (signaling) that may be generated by some applications or services. Network Control Traffic may be split into two service classes:

- o Network Control, and
- o Operations Administration and Management (OAM)

4.1.1. Network Control Protocols

The Network Control service class is used for transmitting packets between network devices (routers) that require control (routing) information to be exchanged between nodes within the administrative domain as well as across a peering point between different administrative domains. The RECOMMENDED DSCP marking for Network Control is CS6.

Before discussing a mapping recommendation for Network Control traffic marked CS6 DSCP, it is interesting to note a relevant recommendation from [RFC4594] pertaining to traffic marked CS7 DSCP: in [RFC4594] Section 3.1 it is RECOMMENDED that packets marked CS7 DSCP (a codepoint that SHOULD be reserved for future use) be dropped or remarked at the edge of the DiffServ domain.

Following this recommendation, it is RECOMMENDED that all packets marked to DiffServ Codepoints not in use over the wireless network be dropped or remarked at the edge of the DiffServ domain.

It is important to note that the wired-to-wireless edge may or may not equate to the edge of the DiffServ domain; as such, this recommendation may or may not apply at the wired-to-wireless edge.

For example, in most commonly deployed models, the wireless access point represents not only the edge of the DiffServ domain, but also the edge of the network infrastructure itself. As such, and in line with the above recommendation, traffic marked CS7 DSCP SHOULD be dropped or remarked at this edge (as it is typically unused, as CS7 SHOULD be reserved for future use). So too SHOULD Network Control traffic marked CS6 DSCP, considering that only client devices (and no network infrastructure devices) are downstream from the wireless access points in these deployment models. In such cases, no Network Control traffic would be (legitimately) expected to be sent or received from wireless client endpoint devices, and thus this recommendation would apply.

Alternatively, in other deployment models, such as Wi-Fi backhaul, wireless mesh infrastructures, or any other type of wireless AP-to-AP deployments, the wireless access point extends the network infrastructure and thus, typically, the DiffServ domain. In such cases, the above recommendation would not apply, as the wired-to-wireless edge does not represent the edge of the DiffServ domain. Furthermore, as these deployment models require Network Control traffic to be propagated across the wireless network, it is RECOMMENDED to map Network Control traffic marked CS6 to UP 7 (per IEEE 802.11-2012, Section 9.2.4.2, Table 9-1), thereby admitting it to the Voice Access Category (AC_VO).

4.1.2. Operations Administration Management (OAM)

The OAM (Operations, Administration, and Management) service class is RECOMMENDED for OAM&P (Operations, Administration, and Management and Provisioning). The RECOMMENDED DSCP marking for OAM is CS2.

By default, packets marked DSCP CS2 will be mapped to UP 2 and serviced with the Background Access Category (AC_BK). Such servicing is a contradiction to the intent expressed in [RFC4594] Section 3.3. As such, it is RECOMMENDED that a non-default mapping be applied to OAM traffic, such that CS2 DSCP is mapped to UP 0, thereby admitting it to the Best Effort Access Category (AC_BE).

4.2. User Traffic

User traffic is defined as packet flows between different users or subscribers. It is the traffic that is sent to or from end-terminals and that supports a very wide variety of applications and services. Network administrators can categorize their applications according to

the type of behavior that they require and MAY choose to support all or a subset of the defined service classes.

4.2.1. Telephony

The Telephony service class is RECOMMENDED for applications that require real-time, very low delay, very low jitter, and very low packet loss for relatively constant-rate traffic sources (inelastic traffic sources). This service class SHOULD be used for IP telephony service. The fundamental service offered to traffic in the Telephony service class is minimum jitter, delay, and packet loss service up to a specified upper bound. The RECOMMENDED DSCP marking for Telephony is EF.

Traffic marked to DSCP EF will map by default to UP 5, and thus to the Video Access Category (AC_VI), rather than to the Voice Access Category (AC_VO), for which it is intended. Therefore, a non-default DSCP-to-UP mapping is RECOMMENDED, such that EF DSCP is mapped to UP 6, thereby admitting it into the Voice Access Category (AC_VO).

Similarly, the [RFC5865] VOICE-ADMIT DSCP (44/101100) is RECOMMENDED to be mapped to UP 6, thereby admitting it also into the Voice Access Category (AC_VO).

4.2.2. Signaling

The Signaling service class is RECOMMENDED for delay-sensitive client-server (traditional telephony) and peer-to-peer application signaling. Telephony signaling includes signaling between IP phone and soft-switch, soft-client and soft-switch, and media gateway and soft-switch as well as peer-to-peer using various protocols. This service class is intended to be used for control of sessions and applications. The RECOMMENDED DSCP marking for Signaling is CS5.

While Signaling is RECOMMENDED to receive a superior level of service relative to the default class (i.e. AC_BE), it does not require the highest level of service (i.e. AC_VO). This leaves only the Video Access Category (AC_VI), which it will map to by default. Therefore it is RECOMMENDED to map Signaling traffic marked CS5 DSCP to UP 5, thereby admitting it to the Video Access Category (AC_VI).

Note: Signaling traffic is not control plane traffic from the perspective of the network (but rather is data plane traffic); as such, it does not merit provisioning in the Network Control service class (marked CS6 and mapped to UP 6). However, Signaling traffic is control-plane traffic from the perspective of the voice/video telephony overlay-infrastructure. As such, Signaling should be treated with preferential servicing vs. other data plane flows. One

way this may be achieved in certain WLAN deployments is by mapping Signaling traffic marked CS5 to UP 5 (as recommended above). To illustrate: IEEE 802.11-2012 displays a reference implementation model in Figure 9-19 which depicts four transmit queues, one per access category. In practical implementation, however, it is common for WLAN network equipment vendors to actually implement dedicated transmit queues on a per-UP basis, which are then dequeued into their associated access category in a preferred (or even strict priority manner). For example, (and specific to this point): it is common for vendors to dequeue UP 5 ahead of UP 4 to the hardware performing the EDCA function (EDCAF) for the Video Access Category (AC_VI). As such, Signaling traffic may benefit from such treatment vs. other video flows in the same access category (as well as vs. data flows in the Best Effort and Background Access Categories) due to this differentiation in servicing under such implementations.

4.2.3. Multimedia Conferencing

The Multimedia Conferencing service class is RECOMMENDED for applications that require real-time service for rate-adaptive traffic. The RECOMMENDED DSCP markings for Multimedia Conferencing are AF41, AF42 and AF43.

The primary media type typically carried within the Multimedia Conferencing service class is video; as such, it is RECOMMENDED to map this class into the Video Access Category (which it does by default). Specifically, it is RECOMMENDED to map AF41, AF42 and AF43 to UP 4, thereby admitting Multimedia Conferencing into the Video Access Category (AC_VI).

4.2.4. Real-Time Interactive

The Real-Time Interactive traffic class is RECOMMENDED for applications that require low loss and jitter and very low delay for variable rate inelastic traffic sources. Such applications may include inelastic video-conferencing applications, but may also include gaming applications (as pointed out in [RFC4594] Sections 2.1 through 2.3, and Section 4.4). The RECOMMENDED DSCP marking for Real-Time Interactive traffic is CS4.

The primary media type typically carried within the Real-Time Interactive service class is video; as such, it is RECOMMENDED to map this class into the Video Access Category (which it does by default). Specifically, it is RECOMMENDED to map CS4 to UP 4, thereby admitting Real-Time Interactive traffic into the Video Access Category (AC_VI).

4.2.5. Multimedia-Streaming

The Multimedia Streaming service class is RECOMMENDED for applications that require near-real-time packet forwarding of variable rate elastic traffic sources. Typically these flows are unidirectional. The RECOMMENDED DSCP markings for Multimedia Streaming are AF31, AF32 and AF33.

The primary media type typically carried within the Multimedia Streaming service class is video; as such, it is RECOMMENDED to map this class into the Video Access Category. Specifically, it is RECOMMENDED to map AF31, AF32 and AF33 to UP 4, thereby admitting Multimedia Streaming into the Video Access Category (AC_VI).

4.2.6. Broadcast Video

The Broadcast Video service class is RECOMMENDED for applications that require near-real-time packet forwarding with very low packet loss of constant rate and variable rate inelastic traffic sources. Typically these flows are unidirectional. The RECOMMENDED DSCP marking for Broadcast Video is CS3.

As directly implied by the name, the primary media type typically carried within the Broadcast Video service class is video; as such, it is RECOMMENDED to map this class into the Video Access Category. Specifically, it is RECOMMENDED to map CS4 to UP 4, thereby admitting Broadcast Video into the Video Access Category (AC_VI).

4.2.7. Low-Latency Data

The Low-Latency Data service class is RECOMMENDED for elastic and time-sensitive data applications, often of a transactional nature, where a user is waiting for a response via the network in order to continue with a task at hand. As such, these flows may be considered foreground traffic, with delays or drops to such traffic directly impacting user-productivity. The RECOMMENDED DSCP markings for Low-Latency Data are AF21, AF22 and AF23.

In line with the recommendations made in Section 4.2.2, mapping Low-Latency Data to UP 3 may allow such to receive a superior level of service via transmit queues servicing the EDCAF hardware for the Best Effort Access Category (AC_BE). Therefore it is RECOMMENDED to map Low-Latency Data traffic marked AF2x DSCP to UP 3, thereby admitting it to the Best Effort Access Category (AC_BE).

4.2.8. High-Throughput Data

The High-Throughput Data service class is RECOMMENDED for elastic applications that require timely packet forwarding of variable rate traffic sources and, more specifically, is configured to provide efficient, yet constrained (when necessary) throughput for TCP longer-lived flows. These flows are typically non-user-interactive. Per [RFC4594]-Section 4.8, it can be assumed that this class will consume any available bandwidth and that packets traversing congested links may experience higher queuing delays or packet loss. It is also assumed that this traffic is elastic and responds dynamically to packet loss. The RECOMMENDED DSCP markings for High-Throughput Data are AF11, AF12 and AF13.

Unfortunately, there really is no corresponding fit for the High-Throughput Data traffic class within the constrained 4 Access Category 802.11 model. If the High-Throughput Data traffic class is assigned to the Best Effort Access Category (AC_BE), then it would contend with Low-Latency Data (while [RFC4594] recommends a distinction in servicing between these traffic classes) as well as with the default traffic class; alternatively, if it is assigned to the Background Access Category (AC_BK), then it would receive a less-than-best-effort service and contend with Low-Priority Data (as discussed in Section 4.2.10).

As such, since there is no directly corresponding fit for the High-Throughput Data service class within the 802.11 model, it is generally RECOMMENDED to map High-Throughput Data to UP 0, thereby admitting it to the Best Effort Access Category (AC_BE).

4.2.9. Standard Service Class

The Standard service class is RECOMMENDED for traffic that has not been classified into one of the other supported forwarding service classes in the DiffServ network domain. This service class provides the Internet's "best-effort" forwarding behavior. The RECOMMENDED DSCP marking for the Standard Service Class is DF.

The Standard Service Class loosely corresponds to the 802.11 Best Effort Access Category (AC_BK) and therefore it is RECOMMENDED to map Standard Service Class traffic marked DF DSCP to UP 0, thereby admitting it to the Best Effort Access Category (AC_BE).

4.2.10. Low-Priority Data

The Low-Priority Data service class serves applications that the user is willing to accept service without guarantees. This service class is specified in [RFC3662].

The Low-Priority Data service class loosely corresponds to the 802.11 Background Access Category (AC_BK) and therefore it is RECOMMENDED to map Low-Priority Data traffic marked CS1 DSCP to UP 1, thereby admitting it to the Background Access Category (AC_BK).

4.3. DSCP-to-UP Mapping Recommendations Summary

Figure 1 summarizes the [RFC4594] DSCP marking recommendations mapped to IEEE 802.11 UP and access categories applied in the downstream direction (from wired-to-wireless networks).

IETF DiffServ Service Class	PHB	Reference RFC	IEEE 802.11 User Priority	Access Category
Network Control	CS6	RFC2474	(See Section 4.1.1)	
Telephony	EF	RFC3246	6	AC_VO (Voice)
VOICE-ADMIT	VOICE- ADMIT	RFC5865	6	AC_VO (Voice)
Signaling	CS5	RFC2474	5	AC_VI (Video)
Multimedia Conferencing	AF41 AF42 AF43	RFC2597	4	AC_VI (Video)
Real-Time Interactive	CS4	RFC2474	4	AC_VI (Video)
Multimedia Streaming	AF31 AF32 AF33	RFC2597	4	AC_VI (Video)
Broadcast Video	CS3	RFC2474	4	AC_VI (Video)
Low- Latency Data	AF21 AF22 AF23	RFC2597	3	AC_BE (Best Effort)
OAM	CS2	RFC2474	0	AC_BE (Best Effort)
High- Throughput Data	AF11 AF12 AF13	RFC2597	0	AC_BE (Best Effort)
Standard	DF	RFC2474	0	AC_BE (Best Effort)
Low-Priority Data	CS1	RFC3662	1	AC_BK (Background)

Figure 1: Summary of Downstream DSCP to IEEE 802.11 UP and AC Mapping Recommendations

5. Upstream Mapping Recommendations

In the upstream direction, there are three types of mapping that may occur:

- o DSCP-to-UP mapping within the wireless client operating system
- o UP-to-DSCP mapping at the wireless access point
- o DSCP-Trust at the wireless access point

5.1. Upstream DSCP-to-UP Mapping within the Wireless Client Operating System

Some operating systems on wireless client devices utilize a similar default DSCP-to-UP mapping scheme as described in Section 2.1. As such, this can lead to the same conflicts as described in that section, but in the upstream direction.

Therefore, to improve on these default mappings, and to achieve parity and consistency with downstream QoS, it is RECOMMENDED that such wireless client operating systems utilize instead the same DSCP-to-UP mapping recommendations presented in Section 4 and/or fully customizable UP markings.

5.2. UP-to-DSCP Mapping at the Wireless Access Point

UP-to-DSCP mapping generates a DSCP value for the IP packet (either an unencapsulated IP packet or an IP packet encapsulated within a tunneling protocol such as CAPWAP - and destined towards a wireless LAN controller for decapsulation and forwarding) from the Layer 2 IEEE UP markings of the wireless frame.

It should be noted that any explicit remarking policy to be performed on such a packet only takes place at the nearest classification and marking policy enforcement point, which may be:

- o At the wireless access point
- o At the wired network switch port
- o At the wireless LAN controller

As such, UP-to-DSCP mapping allows for wireless L2 markings to affect the QoS treatment of a packet over the wired IP network (that is, until the packet reaches the nearest classification and marking policy enforcement point).

It should be noted that nowhere in the IEEE 802.11 specifications is there an intent expressed for 802.11 UP to be used to influence QoS treatment over wired IP networks. Furthermore, both [RFC2474] and [RFC2475] allow for the host to set DSCP markings for QoS treatment over IP networks. Therefore, it is NOT RECOMMENDED that wireless access points trust UP markings as set by these hosts and subsequently perform a UP-to-DSCP mapping in the upstream direction, but rather, if wireless host markings are to be trusted (as per business requirements, technical constraints and administrative preference), then it is RECOMMENDED to trust the DSCP markings set by these wireless hosts.

5.3. DSCP-Trust at the Wireless Access Point

On wireless access points that can trust DSCP markings of packets encapsulated within wireless frames it is RECOMMENDED to trust DSCP markings in the upstream direction, for the following reasons:

- o [RFC2474] and [RFC2475] allow for hosts to set DSCP markings to achieve an end-to-end differentiated service
- o IEEE 802.11 does not specify that UP markings are to be used to affect QoS treatment over wired IP networks
- o Most wireless device operating systems generate UP values by the same method as described in Section 3.1 (i.e. by using the 3 MSB of the encapsulated 6-bit DSCP); then, at the access point, these 3-bit mappings are converted back into DSCP values, either by the default operation described in Section 3.2 or by a customized mapping as described in Section 4; in either case, information is lost in the transitions from 6-bit marking to 3-bit marking and then back to 6-bit marking; trusting the encapsulated DSCP prevents this loss of information
- o A practical implementation benefit is also realized by trusting the DSCP set by wireless client devices, as enabling applications to mark DSCP is much more prevalent and accessible to programmers of wireless applications vis-a-vis trying to explicitly set UP values, which requires special hooks into the wireless device operating system and/or hardware device drivers, many of which (at the time of writing) have little or no resources to support such functionality

6. Appendix: IEEE 802.11 QoS Overview

QoS is enabled on wireless networks by means of the Hybrid Coordination Function (HCF). To give better context to the

enhancements in HCF that enable QoS, it may be helpful to begin with a review of the original Distributed Coordination Function (DCF).

6.1. Distributed Coordination Function (DCF)

As has been noted, the Wi-Fi medium is a shared medium, with each station—including the wireless access point—contending for the medium on equal terms. As such, it shares the same challenge as any other shared medium in requiring a mechanism to prevent (or avoid) collisions which can occur when two (or more) stations attempt simultaneous transmission.

The IEEE Ethernet working group solved this challenge by implementing a Carrier Sense Multiple Access/Collision Detection (CSMA/CD) mechanism that could detect collisions over the shared physical cable (as collisions could be detected as reflected energy pulses over the physical wire). Once a collision was detected, then a pre-defined set of rules was invoked that required stations to back off and wait random periods of time before re-attempting transmission. While CSMA/CD improved the usage of Ethernet as a shared medium, it should be noted the ultimate solution to solving Ethernet collisions was the advance of switching technologies, which treated each Ethernet cable as a dedicated collision domain.

However, unlike Ethernet (which uses physical cables), collisions cannot be directly detected over the wireless medium, as RF energy is radiated over the air and colliding bursts are not necessarily reflected back to the transmitting stations. Therefore, a different mechanism is required for this medium.

As such, the IEEE modified the CSMA/CD mechanism to adapt it to wireless networks to provide Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA). The original CSMA/CA mechanism used in 802.11 was the Distributed Coordination Function. DCF is a timer-based system that leverages three key sets of timers, the slot time, interframe spaces and contention windows.

6.1.1. Slot Time

The slot time is the basic unit of time measure for both DCF and HCF, on which all other timers are based. The slot time duration varies with the different generations of data-rates and performances described by the 802.11 standard. For example, the IEEE 802.11-2012 standard specifies the slot time to be 20 us (IEEE 802.11-2012 Table 16-2) for legacy implementations (such as 802.11b, supporting 1, 2, 5.5 and 11 Mbps data rates), while newer implementations (including 802.11g, 80.11a, 802.11n and 802.11ac, supporting data

rates from 500 Mbps to over 1 Gbps) define a shorter slot time of 9 us (IEEE 802.11-2012, Section 18.4.4, Table 18-17).

6.1.2. Interframe Spaces

The time interval between frames that are transmitted over the air is called the Interframe Space (IFS). Several IFS are defined in 802.11, with the two most relevant to DCF being the Short Interframe Space (SIFS) and the DCF Interframe Space (DIFS).

The SIFS is the amount of time in microseconds required for a wireless interface to process a received RF signal and its associated 802.11 frame and to generate a response frame. Like slot times, the SIFS can vary according to the performance implementation of the 802.11 standard. The SIFS for 802.11a, 802.11n and 802.11ac (in 5 Ghz) is 16 us (IEEE 802.11-2012, Section 18.4.4, Table 18-17).

Additionally, a station must sense the status of the wireless medium before transmitting. If it finds that the medium is continuously idle for the duration of a DIFS, then it is permitted to attempt transmission of a frame (after waiting an additional random backoff period, as will be discussed in the next section). If the channel is found busy during the DIFS interval, the station must defer its transmission until the medium is found idle for the duration of a DIFS interval. The DIFS is calculated as:

$$\text{DIFS} = \text{SIFS} + (2 * \text{Slot time})$$

However, if all stations waited only a fixed amount of time before attempting transmission then collisions would be frequent. To offset this, each station must wait, not only a fixed amount of time (the DIFS) but also a random amount of time (the random backoff) prior to transmission. The range of the generated random backoff timer is bounded by the Contention Window.

6.1.3. Contention Windows

Contention windows bound the range of the generated random backoff timer that each station must wait (in addition to the DIFS) before attempting transmission. The initial range is set between 0 and the Contention Window minimum value (CWmin), inclusive. The CWmin for DCF (in 5 GHz) is specified as 15 slot times (IEEE 802.11- 2012, Section 18.4.4, Table 18-17).

However, it is possible that two (or more) stations happen to pick the exact same random value within this range. If this happens then a collision will occur. At this point, the stations effectively begin the process again, waiting a DIFS and generate a new random

backoff value. However, a key difference is that for this subsequent attempt, the Contention Window approximatively doubles in size (thus exponentially increasing the range of the random value). This process repeats as often as necessary if collisions continue to occur, until the maximum Contention Window size (CWmax) is reached. The CWmax for DCF is specified as 1023 slot times (IEEE 802.11-2012, Section 18.4.4, Table 18-17).

At this point, transmission attempts may still continue (until some other pre-defined limit is reached), but the Contention Window sizes are fixed at the CWmax value.

Incidentally it may be observed that a significant amount of jitter can be introduced by this contention process for wireless transmission access. For example, the incremental transmission delay of 1023 slot times (CWmax) using 9 us slot times may be as high as 9 ms of jitter per attempt. And as previously noted, multiple attempts can be made at CWmax.

6.2. Hybrid Coordination Function (HCF)

Therefore, as can be seen from the preceding description of DCF, there is no preferential treatment of one station over another when contending for the shared wireless media; nor is there any preferential treatment of one type of traffic over another during the same contention process. To support the latter requirement, the IEEE enhanced DCF in 2005 to support QoS, specifying HCF in 802.11, which was integrated into the main 802.11 standard in 2007.

6.2.1. User Priority (UP)

One of the key changes to the 802.11 frame format is the inclusion of a QoS Control field, with 3 bits dedicated for QoS markings. These bits are referred to the User Priority (UP) bits and these support eight distinct marking values: 0-7, inclusive.

While such markings allow for frame differentiation, these alone do not directly affect over-the-air treatment. Rather it is the non-configurable and standard-specified mapping of UP markings to 802.11 Access Categories (AC) that generate differentiated treatment over wireless media.

6.2.2. Access Category (AC)

Pairs of UP values are mapped to four defined access categories that correspondingly specify different treatments of frames over the air. These access categories (in order of relative priority from the top

down) and their corresponding UP mappings are shown in Figure 2 (adapted from IEEE 802.11-2012, Section 9.2.4.2, Table 9-1).

User Priority	Access Category	Designative (informative)
7	AC_VO	Voice
6	AC_VO	Voice
5	AC_VI	Video
4	AC_VI	Video
3	AC_BE	Best Effort
0	AC_BE	Best Effort
2	AC_BK	Background
1	AC_BK	Background

Figure 2: IEEE 802.11 Access Categories and User Priority Mappings

The manner in which these four access categories achieve differentiated service over-the-air is primarily by tuning the fixed and random timers that stations have to wait before sending their respective types of traffic, as will be discussed next.

6.2.3. Arbitration Inter-Frame Space (AIFS)

As previously mentioned, each station must wait a fixed amount of time to ensure the air is clear before attempting transmission. With DCF, the DIFS is constant for all types of traffic. However, with 802.11 the fixed amount of time that a station has to wait will depend on the access category and is referred to as an Arbitration Interframe Space (AIFS). AIFS are defined in slot times and the AIFS per access category are shown in Figure 3 (adapted from IEEE 802.11-2012, Section 8.4.2.31, Table 8-105).

Access Category	Designative (informative)	AIFS (slot times)
AC_VO	Voice	2
AC_VI	Video	2
AC_BE	Best Effort	3
AC_BK	Background	7

Figure 3: Arbitration Interframe Spaces by Access Category

6.2.4. Access Category Contention Windows (CW)

Not only is the fixed amount of time that a station has to wait skewed according to 802.11 access category, but so are the relative sizes of the Contention Windows that bound the random backoff timers, as shown in Figure 4 (adapted from IEEE 802.11-2012, Section 8.4.2.31, Table 8-105).

Access Category	Designative (informative)	CWmin (slot times)	CWmax (slot times)
AC_VO	Voice	3	7
AC_VI	Video	7	15
AC_BE	Best Effort	15	1023
AC_BK	Background	15	1023

Figure 4: Contention Window Sizes by Access Category

When the fixed and randomly generated timers are added together on a per access category basis, then traffic assigned to the Voice Access Category (i.e. traffic marked to UP 6 or 7) will receive a statistically superior service relative to traffic assigned to the Video Access Category (i.e. traffic marked UP 5 and 4), which, in turn, will receive a statistically superior service relative to traffic assigned to the Best Effort Access Category traffic (i.e.

traffic marked UP 3 and 0), which finally will receive a statistically superior service relative to traffic assigned to the Background Access Category traffic (i.e. traffic marked to UP 2 and 1).

6.3. IEEE 802.11u QoS Map Set

IEEE 802.11u [IEEE.802-11u.2011] is proposed addendum to the IEEE 802.11 standard which includes, among other enhancements, a mechanism by which wireless access points can communicate DSCP to/from UP mappings that have been configured on the wired IP network. Specifically, a QoS Map Set information element (described in IEEE 802.11u-2011 Section 7.3.2.95) is transmitted from an AP to a wireless endpoint device in an association / re-association Response frame (or within a special QoS Map Configure frame). The purpose of the QoS Map Set information element is to provide the mapping of higher layer Quality of Service constructs (i.e. DSCP) to User Priorities so that a wireless endpoint device (that supports this function and is administratively configured to enable it) can perform corresponding DSCP-to-UP mapping within the device (i.e. between applications and the operating system / wireless network interface hardware drivers) to align with what the APs are mapping in the downstream direction, so as to achieve consistent end-to-end QoS.

The QoS Map Set information element includes two key components:

- 1) each of the eight UP values (0-7) are associated with a range of DSCP values, and
- 2) (up to 21) exceptions from these range-based DSCP to/from UP mapping associations may be optionally and explicitly specified.

In line with the recommendations put forward in this document, the following recommendations apply when the this QoS Map Set information element is enabled:

- 1) each of the eight UP values (0-7) are RECOMMENDED to be mapped to DSCP 0 (as a baseline, so as to meet the recommendation made in Section 4.1.1 (that packets marked to unused DiffServ Codepoints be remarked at the edge of the DiffServ domain), and
- 2) (up to 21) exceptions from this baseline mapping are RECOMMENDED to be made in line with Section 4.3, to correspond to the DiffServ Codepoints that are in use over the IP network.

7. IANA Considerations

This memo asks the IANA for no new parameters.

8. Security Considerations

The recommendation offered in Section 4.1.1 (of dropping or remarking packets marked with DiffServ Codepoints not in use at the edge of the DiffServ domain) is to address a Denial-of-Service attack vector that exists at wired-to-wireless edges due to the requirement of trusting traffic markings to ensure end-to-end QoS. For example, consider a malicious user flooding traffic marked CS7 or CS6 DSCP toward the WLAN. These codepoints would map by default to UP 7 and UP 6 (respectively), both of which would be assigned to the Voice Access Category (AC_VO). Such a flood could cause a Denial-of-Service to wireless voice applications.

8.1. Privacy Considerations

9. Acknowledgements

The authors wish to thank TSVWG reviewers.

The authors acknowledge a great many inputs, notably from Jerome Henry, David Kloper, Mark Montanez, Glen Lavers, Michael Fingleton, Sarav Radhakrishnan, Karthik Dakshinamoorthy, Simone Arena, Ranga Marathe, Ramachandra Murthy and many others.

10. References

10.1. Normative References

- [IEEE.802-11.2012] "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", IEEE Standard 802.11, 2012, <<http://standards.ieee.org/getieee802/download/802.11-2012.pdf>>.
- [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>>.
- [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>>.
- [RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services", RFC 3662, DOI 10.17487/RFC3662, December 2003, <<http://www.rfc-editor.org/info/rfc3662>>.
- [RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <<http://www.rfc-editor.org/info/rfc4594>>.
- [RFC5865] Baker, F., Polk, J., and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, DOI 10.17487/RFC5865, May 2010, <<http://www.rfc-editor.org/info/rfc5865>>.

10.2. Informative References

- [I-D.ietf-tsvwg-diffserv-intercon]
Geib, R. and D. Black, "Diffserv-Interconnection classes and practice", draft-ietf-tsvwg-diffserv-intercon-06 (work in progress), June 2016.

- [IEEE.802-11u.2011]
"Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", IEEE Standard 802.11, 2011, <<http://standards.ieee.org/getieee802/download/802.11u-2011.pdf>>.
- [RFC5127] Chan, K., Babiarz, J., and F. Baker, "Aggregation of Diffserv Service Classes", RFC 5127, DOI 10.17487/RFC5127, February 2008, <<http://www.rfc-editor.org/info/rfc5127>>.
- [RFC7561] Kaippallimalil, J., Pazhyannur, R., and P. Yegani, "Mapping Quality of Service (QoS) Procedures of Proxy Mobile IPv6 (PMIPv6) and WLAN", RFC 7561, DOI 10.17487/RFC7561, June 2015, <<http://www.rfc-editor.org/info/rfc7561>>.

Appendix A. Change Log

Initial Version: July 2015

Authors' Addresses

Tim Szigeti
Cisco Systems
Vancouver, British Columbia V7X 1J1
Canada

Email: szigeti@cisco.com

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

Email: fred@cisco.com

Transport Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: June 21, 2018

T. Szigeti
J. Henry
Cisco Systems
F. Baker
December 18, 2017

Diffserv to IEEE 802.11 Mapping
draft-ietf-tsvwg-ieee-802-11-11

Abstract

As internet traffic is increasingly sourced-from and destined-to wireless endpoints, it is crucial that Quality of Service be aligned between wired and wireless networks; however, this is not always the case by default. This document specifies a set of Differentiated Services Code Point (DSCP) to IEEE 802.11 User Priority (UP) mappings to reconcile the marking recommendations offered by the IETF and the IEEE so as to maintain consistent QoS treatment between wired and IEEE 802.11 wireless networks.

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 June 21, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Related work	3
1.2. Interaction with RFC 7561	4
1.3. Applicability Statement	4
1.4. Document Organization	5
1.5. Requirements Language	5
1.6. Terminology Used in this Document	6
2. Service Comparison and Default Interoperation of Diffserv and IEEE 802.11	9
2.1. Diffserv Domain Boundaries	9
2.2. EDCF Queuing	10
2.3. Default DSCP-to-UP Mappings and Conflicts	10
2.4. Default UP-to-DSCP Mappings and Conflicts	11
3. Wireless Device Marking and Mapping Capability Recommendations	13
4. DSCP-to-UP Mapping Recommendations	13
4.1. Network Control Traffic	14
4.1.1. Network Control Protocols	14
4.1.2. Operations Administration Management (OAM)	15
4.2. User Traffic	15
4.2.1. Telephony	15
4.2.2. Signaling	16
4.2.3. Multimedia Conferencing	16
4.2.4. Real-Time Interactive	17
4.2.5. Multimedia-Streaming	17
4.2.6. Broadcast Video	17
4.2.7. Low-Latency Data	18
4.2.8. High-Throughput Data	18
4.2.9. Standard Service Class	19
4.2.10. Low-Priority Data	19
4.3. DSCP-to-UP Mapping Recommendations Summary	20
5. Upstream Mapping and Marking Recommendations	21
5.1. Upstream DSCP-to-UP Mapping within the Wireless Client Operating System	22
5.2. Upstream UP-to-DSCP Mapping at the Wireless Access Point	22
5.3. Upstream DSCP-Passthrough at the Wireless Access Point	23
5.4. Upstream DSCP Marking at the Wireless Access Point	24
6. IEEE 802.11 QoS Overview	24
6.1. Distributed Coordination Function (DCF)	24
6.1.1. Slot Time	25
6.1.2. Interframe Spaces	25

6.1.3. Contention Windows	26
6.2. Hybrid Coordination Function (HCF)	27
6.2.1. User Priority (UP)	27
6.2.2. Access Category (AC)	27
6.2.3. Arbitration Inter-Frame Space (AIFS)	28
6.2.4. Access Category Contention Windows (CW)	29
6.3. IEEE 802.11u QoS Map Set	30
7. IANA Considerations	31
8. Security Considerations	31
8.1. General QoS Security Recommendations	31
8.2. WLAN QoS Security Recommendations	32
9. Acknowledgements	34
10. References	34
10.1. Normative References	34
10.2. Informative References	35
Appendix A. Change Log	36
Authors' Addresses	36

1. Introduction

IEEE 802.11 [IEEE.802.11-2016] wireless has become the preferred medium for endpoints connecting to business and private networks. However, the wireless medium defined by IEEE 802.11 [IEEE.802.11-2016] presents several design challenges for ensuring end-to-end quality of service. Some of these challenges relate to the nature of the IEEE 802.11 Radio Frequency (RF) medium itself, being a half-duplex and shared medium, while other challenges relate to the fact that the IEEE 802.11 standard is not administered by the same standards body as IP networking standards. While the IEEE has developed tools to enable QoS over wireless networks, little guidance exists on how to maintain consistency of QoS treatment between wired IP and wireless IEEE 802.11 networks. The purpose of this document is to provide such guidance.

1.1. Related work

Several RFCs outline Diffserv QoS recommendations over IP networks, including:

- o [RFC2474] specifies the Diffserv Codepoint Field. This RFC also details Class Selectors, as well as the Default Forwarding (DF) treatment.
- o [RFC2475] defines a Diffserv architecture
- o [RFC3246] specifies the Expedited Forwarding (EF) Per-Hop Behavior (PHB)

- o [RFC2597] specifies the Assured Forwarding (AF) PHB.
- o [RFC3662] specifies a Lower Effort Per-Domain Behavior (PDB)
- o [RFC4594] presents Configuration Guidelines for Diffserv Service Classes
- o [RFC5127] presents the Aggregation of Diffserv Service Classes
- o [RFC5865] specifies a DSCP for Capacity Admitted Traffic

Note: [RFC4594] is intended to be viewed as a framework for supporting Diffserv in any network, including wireless networks; thus, it describes different types of traffic expected in IP networks and provides guidance as to what DSCP marking(s) should be associated with each traffic type. As such, this document draws heavily on [RFC4594], as well as [RFC5127], and [RFC8100].

In turn, the relevant standard for wireless QoS is IEEE 802.11, which is being progressively updated; the current version of which (at the time of writing) is [IEEE.802.11-2016].

1.2. Interaction with RFC 7561

There is also a recommendation from the Global System for Mobile Communications Association (GSMA) on DSCP to UP Mapping for IP Packet eXchange (IPX), specifically their Guidelines for IPX Provider networks [GSMA-IPX_Guidelines]. These GSMA Guidelines were developed without reference to existing IETF specifications for various services, referenced in Section 1.1. In turn, [RFC7561] was written based on these GSMA Guidelines, as explicitly called out in [RFC7561] Section 4.2. Thus, [RFC7561] conflicts with the overall Diffserv traffic-conditioning service plan, both in the services specified and the code points specified for them. As such, these two plans cannot be normalized. Rather, as discussed in [RFC2474] Section 2, the two domains (IEEE 802.11 and GSMA) are different Differentiated Services Domains separated by a Differentiated Services Boundary. At that boundary, code points from one domain are translated to code points for the other, and maybe to Default (zero) if there is no corresponding service to translate to.

1.3. Applicability Statement

This document is applicable to the use of Differentiated Services that interconnect with IEEE 802.11 wireless LANs (referred to as Wi-Fi, throughout this document, for simplicity). These guidelines are applicable whether the wireless access points (APs) are deployed in an autonomous manner, managed by (centralized or distributed) WLAN

controllers or some hybrid deployment option. This is because in all these cases, the wireless access point is the bridge between wired and wireless media.

This document applies to IP networks using WiFi infrastructure at the link layer. Such networks typically include wired LANs with wireless access points at their edges, however, such networks can also include Wi-Fi backhaul, wireless mesh solutions or any other type of AP-to-AP wireless network that extends the wired network infrastructure.

1.4. Document Organization

This document is organized as follows:

- o Section 1 introduces the wired-to-wireless QoS challenge, references related work, outlines the organization of the document, and specifies both the requirements language and the terminology used in this document.
- o Section 2 begins the discussion with a comparison of IETF Diffserv QoS and Wi-Fi QoS standards and highlights discrepancies between these that require reconciliation.
- o Section 3 presents the marking and mapping capabilities that wireless access points and wireless endpoint devices are recommended to support.
- o Section 4 presents DSCP-to-UP mapping recommendations for each of the [RFC4594] service classes, which are primarily applicable in the downstream (wired-to-wireless) direction.
- o Section 5, in turn, considers upstream (wireless-to-wired) QoS options, their respective merits and recommendations.
- o Section 6 (in the form of an Appendix) presents a brief overview of how QoS is achieved over IEEE 802.11 wireless networks, given the shared, half-duplex nature of the wireless medium.
- o Section 7 on notes IANA considerations
- o Section 8 presents security considerations relative to DSCP-to-UP, UP-to-DSCP mapping and remarking

1.5. Requirements Language

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.

1.6. Terminology Used in this Document

Key terminology used in this document includes:

AC: Access Category. A label for the common set of enhanced distributed channel access (EDCA) parameters that are used by a quality-of-service (QoS) station (STA) to contend for the channel in order to transmit medium access control (MAC) service data units (MSDUs) with certain priorities. [IEEE.802.11-2016] Section 3.2.

AIFS: Arbitration Interframe Space. Interframe space used by QoS stations before transmission of data and other frame types defined by [IEEE.802.11-2016] Section 10.3.2.3.6.

AP: Access Point. An entity that contains one station (STA) and provides access to the distribution services, via the wireless medium (WM) for associated STAs. An AP comprises a STA and a distribution system access function (DSAF) [IEEE.802.11-2016] Section 3.1.

BSS: Basic Service Set. Informally, a wireless cell; formally, a set of stations that have successfully synchronized using the JOIN service primitives and one STA that has used the START primitive. Alternatively, a set of STAs that have used the START primitive specifying matching mesh profiles where the match of the mesh profiles has been verified via the scanning procedure. Membership in a BSS does not imply that wireless communication with all other members of the BSS is possible. Defined in [IEEE.802.11-2016] Section 3.1.

Contention Window: See CW.

CSMA/CA: Carrier Sense Multiple Access with Collision Avoidance. A media access control method in which carrier sensing is used, but nodes attempt to avoid collisions by transmitting only when the channel is sensed to be "idle". When these do transmit, nodes transmit their packet data in its entirety.

CSMA/CD: Carrier Sense Multiple Access with Collision Detection. A media access control method (used most notably in early Ethernet technology) for local area networking. It uses a carrier-sensing scheme in which a transmitting station detects collisions by sensing transmissions from other stations while transmitting a frame. When this collision condition is detected, the station

stops transmitting that frame, transmits a jam signal, and then waits for a random time interval before trying to resend the frame.

CW: Contention Window. Limits a CWMin and CWMax, from which a random backoff is computed.

CWMax: Contention Window Maximum. The maximum value (in unit of Slot Time) that a contention window can take.

CWMin: Contention Window Minimum. The minimum value that a contention window can take.

DCF: Distributed Coordinated Function. A class of coordination function where the same coordination function logic is active in every station (STA) in the basic service set (BSS) whenever the network is in operation.

DIFS: Distributed (Coordination Function) Interframe Space. A unit of time during which the medium has to be detected as idle before a station should attempt to send frames, as per [IEEE.802.11-2016] Section 10.3.2.3.5.

DSCP: Differentiated Service Code Point [RFC2474] and [RFC2475]. The DSCP is carried in the first 6 bits of the IPv4 and IPv6 Type of Service (TOS) Byte (the remaining 2 bits are used for IP Explicit Congestion Notification [RFC3168]).

EIFS: Extended Interframe Space. A unit of time that a station has to defer before transmitting a frame if the previous frame contained an error, as per [IEEE.802.11-2016] Section 10.3.2.3.7.

HCF: Hybrid Coordination Function A coordination function that combines and enhances aspects of the contention based and contention free access methods to provide quality-of-service (QoS) stations (STAs) with prioritized and parameterized QoS access to the wireless medium (WM), while continuing to support non-QoS STAs for best-effort transfer. [IEEE.802.11-2016] Section 3.1.

IFS: Interframe Space. Period of silence between transmissions over 802.11 networks. [IEEE.802.11-2016] describes several types of Interframe Spaces.

Random Backoff Timer: A pseudorandom integer period of time (in units of Slot Time) over the interval $(0, CW)$, where CW_{min} is less-than-or-equal-to CW , which in turn is less-than-or-equal-to CW_{max} . Stations desiring to initiate transfer of data frames and-or Management frames using the DCF shall invoke the carrier sense

mechanism to determine the busy-or-idle state of the medium. If the medium is busy, the STA shall defer until the medium is determined to be idle without interruption for a period of time equal to DIFS when the last frame detected on the medium was received correctly, or after the medium is determined to be idle without interruption for a period of time equal to EIFS when the last frame detected on the medium was not received correctly. After this DIFS or EIFS medium idle time, the STA shall then generate a random backoff period for an additional deferral time before transmitting. [IEEE.802.11-2016] Section 10.3.3.

RF: Radio Frequency.

SIFS: Short Interframe Space. An IFS used before transmission of specific frames as defined in [IEEE.802.11-2016] Section 10.3.2.3.3.

Slot Time: A unit of time used to count time intervals in 802.11 networks, and defined in [IEEE.802.11-2016] Section 10.3.2.13.

Trust: From a QoS-perspective, trust refers to the accepting of the QoS markings of a packet by a network device. Trust is typically extended at Layer 3 (by accepting the DSCP), but may also be extended at lower layers, such as at Layer 2 by accepting User Priority markings. For example, if an access point is configured to trust DSCP markings and it receives a packet marked EF, then it would treat the packet with the Expedite Forwarding PHB and propagate the EF marking value (DSCP 46) as it transmits the packet. Alternatively, if a network device is configured to operate in an untrusted manner, then it would remark packets as these entered the device, typically to DF (or to a different marking value at the network administrator's preference). Note: The terms "trusted" and "untrusted" are used extensively in [RFC4594].

UP: User Priority. A value associated with a medium access control (MAC) service data unit (MSDU) that indicates how the MSDU is to be handled. The UP is assigned to an MSDU in the layers above the MAC [IEEE.802.11-2016] Section 3.1. The UP defines a level of priority for the associated frame, on a scale of 0 to 7.

Wi-Fi: An interoperability certification defined by the Wi-Fi Alliance. However, this term is commonly used, including in the present document, to be the equivalent of IEEE 802.11.

Wireless: In the context of this document, "wireless" refers to the media defined in IEEE 802.11 [IEEE.802.11-2016], and not 3G/4G LTE or any other radio telecommunications specification.

2. Service Comparison and Default Interoperation of Diffserv and IEEE 802.11

(Section 6 provides a brief overview of IEEE 802.11 QoS.)

The following comparisons between IEEE 802.11 and Diffserv services should be noted:

- o [IEEE.802.11-2016] does not support an EF PHB service [RFC3246], as it is not possible to assure that a given access category will be serviced with strict priority over another (due to the random element within the contention process)
- o [IEEE.802.11-2016] does not support an AF PHB service [RFC2597], again because it is not possible to assure that a given access category will be serviced with a minimum amount of assured bandwidth (due to the non-deterministic nature of the contention process)
- o [IEEE.802.11-2016] loosely supports a [RFC2474] Default Forwarding service via the Best Effort Access Category (AC_BE)
- o [IEEE.802.11-2016] loosely supports a [RFC3662] Lower Effort PDB service via the Background Access Category (AC_BK)

As such, these high-level considerations should be kept in mind when mapping from Diffserv to [IEEE.802.11-2016] (and vice-versa); however, access points may or may not always be positioned at Diffserv domain boundaries, as will be discussed next.

2.1. Diffserv Domain Boundaries

It is important to recognize that the wired-to-wireless edge may or may not function as an edge of a Diffserv domain or a domain boundary.

In most commonly-deployed WLAN models, the wireless access point represents not only the edge of the Diffserv domain, but also the edge of the network infrastructure itself. As such, only client endpoint devices (and no network infrastructure devices) are downstream from the access points in these deployment models. Note: security considerations and recommendations for hardening such Wifi-at-the-edge deployment models are detailed in Section 8; these recommendations include mapping network control protocols (which are not used downstream from the AP in this deployment model) to UP 0.

Alternatively, in other deployment models, such as Wi-Fi backhaul, wireless mesh infrastructures, wireless AP-to-AP deployments, or in

cases where a Wi-Fi link connects to a device providing service via another technology (e.g. Wi-Fi to Bluetooth or Zigbee router), the wireless access point extends the network infrastructure and thus, typically, the Diffserv domain. In such deployments, both client devices and infrastructure devices may be expected downstream from the access points, and as such network control protocols are RECOMMENDED to be mapped to UP 7 in this deployment model, as is discussed in Section 4.1.1.

Thus, as can be seen from these two examples, the QoS treatment of packets at the access point will depend on the position of the AP in the network infrastructure and on the WLAN deployment model.

However, regardless of the access point being at the Diffserv boundary or not, Diffserv to [IEEE.802.11-2016] (and vice-versa) marking-specific incompatibilities exist that must be reconciled, as will be discussed next.

2.2. EDCF Queuing

[IEEE.802.11-2016] displays a reference implementation queuing model in Figure 10-24, which depicts four transmit queues, one per access category.

However, in practical implementations, it is common for WLAN network equipment vendors to implement dedicated transmit queues on a per-UP (versus a per access category) basis, which are then dequeued into their associated access category in a preferred (or even in a strict priority manner). For example, it is common for vendors to dequeue UP 5 ahead of UP 4 to the hardware performing the EDCA function (EDCAF) for the Video Access Category (AC_VI).

Some of the recommendations made in Section 4 make reference to this common implementation model of queuing per UP.

2.3. Default DSCP-to-UP Mappings and Conflicts

While no explicit guidance is offered in mapping (6-Bit) Layer 3 DSCP values to (3-Bit) Layer 2 markings (such as IEEE 802.1D, 802.1p or 802.11e), a common practice in the networking industry is to map these by what we will refer to as 'Default DSCP-to-UP Mapping' (for lack of a better term), wherein the 3 Most Significant Bits (MSB) of the DSCP are used as the corresponding L2 markings.

Note: There are mappings provided in [IEEE.802.11-2016] Annex V Tables V-1 and V2, but it bears mentioning that these mappings are provided as examples (as opposed to explicit recommendations). Furthermore, some of these mappings do not align with the intent and

recommendations expressed in [RFC4594], as will be discussed in this and the following section (Section 2.4).

However, when this default DSCP-to-UP mapping method is applied to packets marked per [RFC4594] recommendations and destined to 802.11 WLAN clients, it will yield a number of inconsistent QoS mappings, specifically:

- o Voice (EF-101110) will be mapped to UP 5 (101), and treated in the Video Access Category (AC_VI), rather than the Voice Access Category (AC_VO), for which it is intended
- o Multimedia Streaming (AF3-011xx0) will be mapped to UP3 (011) and treated in the Best Effort Access Category (AC_BE), rather than the Video Access Category (AC_VI), for which it is intended
- o Broadcast Video (CS3-011000) will be mapped to UP3 (011) and treated in the Best Effort Access Category (AC_BE), rather than the Video Access Category (AC_VI), for which it is intended
- o OAM traffic (CS2-010000) will be mapped to UP 2 (010) and treated in the Background Access Category (AC_BK), which is not the intent expressed in [RFC4594] for this service class

It should also be noted that while [IEEE.802.11-2016] defines an intended use for each access category through the AC naming convention (for example, UP 6 and UP 7 belong to AC_VO, the Voice Access Category), [IEEE.802.11-2016] does not:

- o define how upper layer markings (such as DSCP) should map to UPs (and hence to ACs)
- o define how UPs should translate to other medium Layer 2 QoS markings
- o strictly restrict each access category to applications reflected in the AC name

2.4. Default UP-to-DSCP Mappings and Conflicts

In the opposite direction of flow (the upstream direction, that is, from wireless-to-wired), many APs use what we will refer to as 'Default UP-to-DSCP Mapping' (for lack of a better term), wherein DSCP values are derived from UP values by multiplying the UP values by 8 (i.e. shifting the 3 UP bits to the left and adding three additional zeros to generate a DSCP value). This derived DSCP value is then used for QoS treatment between the wireless access point and the nearest classification and marking policy enforcement point

(which may be the centralized wireless LAN controller, relatively deep within the network). Alternatively, in the case where there is no other classification and marking policy enforcement point, then this derived DSCP value will be used on the remainder of the Internet path.

It goes without saying that when 6 bits of marking granularity are derived from 3, then information is lost in translation. Servicing differentiation cannot be made for 12 classes of traffic (as recommended in [RFC4594]), but for only 8 (with one of these classes being reserved for future use (i.e. UP 7 which maps to DSCP CS7)).

Such default upstream mapping can also yield several inconsistencies with [RFC4594], including:

- o Mapping UP 6 ([RFC4594] Voice) to CS6, which [RFC4594] recommends for Network Control
- o Mapping UP 4 ([RFC4594] Multimedia Conferencing and/or Real-Time Interactive) to CS4, thus losing the ability to differentiate between these two distinct service classes, as recommended in [RFC4594] Sections 4.3 and 4.4
- o Mapping UP 3 ([RFC4594] Multimedia Streaming and/or Broadcast Video) to CS3, thus losing the ability to differentiate between these two distinct service classes, as recommended in [RFC4594] Sections 4.5 and 4.6
- o Mapping UP 2 ([RFC4594] Low-Latency Data and/or OAM) to CS2, thus losing the ability to differentiate between these two distinct service classes, as recommended in [RFC4594] Sections 4.7 and 3.3, and possibly overwhelming the queues provisioned for OAM (which is typically lower in capacity [being network control traffic], as compared to Low-Latency Data queues [being user traffic])
- o Mapping UP 1 ([RFC4594] High-Throughput Data and/or Low-Priority Data) to CS1, thus losing the ability to differentiate between these two distinct service classes, as recommended in [RFC4594] Sections 4.8 and 4.10, and causing legitimate business-relevant High-Throughput Data to receive a [RFC3662] Lower Effort PDB, for which it is not intended

The following sections address these limitations and concerns in order to reconcile [RFC4594] and [IEEE.802.11-2016]. First downstream (wired-to-wireless) DSCP-to-UP mappings will be aligned and then upstream (wireless-to-wired) models will be addressed.

3. Wireless Device Marking and Mapping Capability Recommendations

This document assumes and RECOMMENDS that all wireless access points (as the interconnects between wired-and-wireless networks) support the ability to:

- o mark DSCP, per Diffserv standards
- o mark UP, per the [IEEE.802.11-2016] standard
- o support fully-configurable mappings between DSCP and UP
- o process DSCP markings set by wireless endpoint devices

This document further assumes and RECOMMENDS that all wireless endpoint devices support the ability to:

- o mark DSCP, per Diffserv standards
- o mark UP, per the [IEEE.802.11-2016] standard
- o support fully-configurable mappings between DSCP (set by applications in software) and UP (set by the operating system and/or wireless network interface hardware drivers)

Having made the assumptions and recommendations above, it bears mentioning while the mappings presented in this document are RECOMMENDED to replace the current common default practices (as discussed in Section 2.3 and Section 2.4), these mapping recommendations are not expected to fit every last deployment model, and as such MAY be overridden by network administrators, as needed.

4. DSCP-to-UP Mapping Recommendations

The following section specifies downstream (wired-to-wireless) mappings between [RFC4594] Configuration Guidelines for Diffserv Service Classes and [IEEE.802.11-2016]. As such, this section draws heavily from [RFC4594], including service class definitions and recommendations.

This section assumes [IEEE.802.11-2016] wireless access points and/or WLAN controllers that support customizable, non-default DSCP-to-UP mapping schemes.

This section also assumes that [IEEE.802.11-2016] access points and endpoint devices differentiate UP markings with corresponding queuing and dequeuing treatments, as described in Section 2.2.

4.1. Network Control Traffic

Network control traffic is defined as packet flows that are essential for stable operation of the administered network [RFC4594] Section 3. Network control traffic is different from user application control (signaling) that may be generated by some applications or services. Network control traffic MAY be split into two service classes:

- o Network Control, and
- o Operations Administration and Management (OAM)

4.1.1. Network Control Protocols

The Network Control service class is used for transmitting packets between network devices (e.g. routers) that require control (routing) information to be exchanged between nodes within the administrative domain, as well as across a peering point between different administrative domains.

[RFC4594] Section 3.2 recommends that Network Control traffic be marked CS6 DSCP. Additionally, as stated in [RFC4594] Section 3.1: "CS7 DSCP value SHOULD be reserved for future use, potentially for future routing or control protocols."

By default (as described in Section 2.3), packets marked DSCP CS7 will be mapped to UP 7 and serviced within the Voice Access Category (AC_VO). This represents the RECOMMENDED mapping for CS7, that is, packets marked to CS7 DSCP are RECOMMENDED to be mapped to UP 7.

However, by default (as described in Section 2.3), packets marked DSCP CS6 will be mapped to UP 6 and serviced within the Voice Access Category (AC_VO); such mapping and servicing is a contradiction to the intent expressed in [RFC4594] Section 3.2. As such, it is RECOMMENDED to map Network Control traffic marked CS6 to UP 7 (per [IEEE.802.11-2016] Section 10.2.4.2, Table 10-1), thereby admitting it to the Voice Access Category (AC_VO), albeit with a marking distinguishing it from (data-plane) voice traffic.

It should be noted that encapsulated routing protocols for encapsulated or overlay networks (e.g., VPN, Network Virtualization Overlays, etc.) are not network control traffic for any physical network at the AP, and hence SHOULD NOT be marked with CS6 in the first place.

Additionally, and as previously noted, the Security Considerations section (Section 8) contains additional recommendations for hardening Wifi-at-the-edge deployment models, where, for example, network

control protocols are not expected to be sent nor received between APs and downstream endpoint client devices.

4.1.2. Operations Administration Management (OAM)

The OAM (Operations, Administration, and Management) service class is recommended for OAM&P (Operations, Administration, and Management and Provisioning). The OAM service class can include network management protocols, such as SNMP, SSH, TFTP, Syslog, etc., as well as network services, such as NTP, DNS, DHCP, etc. [RFC4594] Section 3.3 recommends that OAM traffic be marked CS2 DSCP.

By default (as described in Section 2.3), packets marked DSCP CS2 will be mapped to UP 2 and serviced with the Background Access Category (AC_BK). Such servicing is a contradiction to the intent expressed in [RFC4594] Section 3.3. As such, it is RECOMMENDED that a non-default mapping be applied to OAM traffic, such that CS2 DSCP is mapped to UP 0, thereby admitting it to the Best Effort Access Category (AC_BE).

4.2. User Traffic

User traffic is defined as packet flows between different users or subscribers. It is the traffic that is sent to or from end-terminals and that supports a very wide variety of applications and services [RFC4594] Section 4.

Network administrators can categorize their applications according to the type of behavior that they require and MAY choose to support all or a subset of the defined service classes.

4.2.1. Telephony

The Telephony service class is recommended for applications that require real-time, very low delay, very low jitter, and very low packet loss for relatively constant-rate traffic sources (inelastic traffic sources). This service class SHOULD be used for IP telephony service. The fundamental service offered to traffic in the Telephony service class is minimum jitter, delay, and packet loss service up to a specified upper bound. [RFC4594] Section 4.1 recommends that Telephony traffic be marked EF DSCP.

Traffic marked to DSCP EF will map by default (as described in Section 2.3) to UP 5, and thus to the Video Access Category (AC_VI), rather than to the Voice Access Category (AC_VO), for which it is intended. Therefore, a non-default DSCP-to-UP mapping is RECOMMENDED, such that EF DSCP is mapped to UP 6, thereby admitting it into the Voice Access Category (AC_VO).

Similarly, the [RFC5865] VOICE-ADMIT DSCP (44/101100) is RECOMMENDED to be mapped to UP 6, thereby admitting it also into the Voice Access Category (AC_VO).

4.2.2. Signaling

The Signaling service class is recommended for delay-sensitive client-server (e.g. traditional telephony) and peer-to-peer application signaling. Telephony signaling includes signaling between IP phone and soft-switch, soft-client and soft-switch, and media gateway and soft-switch as well as peer-to-peer using various protocols. This service class is intended to be used for control of sessions and applications. [RFC4594] Section 4.2 recommends that Signaling traffic be marked CS5 DSCP.

While Signaling is recommended to receive a superior level of service relative to the default class (i.e. AC_BE), it does not require the highest level of service (i.e. AC_VO). This leaves only the Video Access Category (AC_VI), which it will map to by default (as described in Section 2.3). Therefore it is RECOMMENDED to map Signaling traffic marked CS5 DSCP to UP 5, thereby admitting it to the Video Access Category (AC_VI).

Note: Signaling traffic is not control plane traffic from the perspective of the network (but rather is data plane traffic); as such, it does not merit provisioning in the Network Control service class (marked CS6 and mapped to UP 6). However, Signaling traffic is control-plane traffic from the perspective of the voice/video telephony overlay-infrastructure. As such, Signaling should be treated with preferential servicing vs. other data plane flows. This may be achieved in common WLAN deployments by mapping Signaling traffic marked CS5 to UP 5. On APs supporting per-UP EDCAF queuing logic (as described in Section 2.2) this will result in preferential treatment for Signaling traffic versus other video flows in the same access category (AC_VI), which are marked to UP 4, as well as preferred treatment over flows in the Best Effort (AC_BE) and Background (AC_BK) access categories.

4.2.3. Multimedia Conferencing

The Multimedia Conferencing service class is recommended for applications that require real-time service for rate-adaptive traffic. [RFC4594] Section 4.3 recommends Multimedia Conferencing traffic be marked AF4x (that is, AF41, AF42 and AF43, according to the rules defined in [RFC2475]).

The primary media type typically carried within the Multimedia Conferencing service class is video; as such, it is RECOMMENDED to

map this class into the Video Access Category, which it does by default (as described in Section 2.3). Specifically, it is RECOMMENDED to map AF41, AF42 and AF43 to UP 4, thereby admitting Multimedia Conferencing into the Video Access Category (AC_VI).

4.2.4. Real-Time Interactive

The Real-Time Interactive service class is recommended for applications that require low loss and jitter and very low delay for variable rate inelastic traffic sources. Such applications may include inelastic video-conferencing applications, but may also include gaming applications (as pointed out in [RFC4594] Sections 2.1 through 2.3, and Section 4.4). [RFC4594] Section 4.4 recommends Real-Time Interactive traffic be marked CS4 DSCP.

The primary media type typically carried within the Real-Time Interactive service class is video; as such, it is RECOMMENDED to map this class into the Video Access Category, which it does by default (as described in Section 2.3). Specifically, it is RECOMMENDED to map CS4 to UP 4, thereby admitting Real-Time Interactive traffic into the Video Access Category (AC_VI).

4.2.5. Multimedia-Streaming

The Multimedia Streaming service class is recommended for applications that require near-real-time packet forwarding of variable rate elastic traffic sources. Typically these flows are unidirectional. [RFC4594] Section 4.5 recommends Multimedia Streaming traffic be marked AF3x (that is, AF31, AF32 and AF33, according to the rules defined in [RFC2475]).

The primary media type typically carried within the Multimedia Streaming service class is video; as such, it is RECOMMENDED to map this class into the Video Access Category, which it will by default (as described in Section 2.3). Specifically, it is RECOMMENDED to map AF31, AF32 and AF33 to UP 4, thereby admitting Multimedia Streaming into the Video Access Category (AC_VI).

4.2.6. Broadcast Video

The Broadcast Video service class is recommended for applications that require near-real-time packet forwarding with very low packet loss of constant rate and variable rate inelastic traffic sources. Typically these flows are unidirectional. [RFC4594] Section 4.6 recommends Broadcast Video traffic be marked CS3 DSCP.

As directly implied by the name, the primary media type typically carried within the Broadcast Video service class is video; as such,

it is RECOMMENDED to map this class into the Video Access Category; however, by default (as described in Section 2.3), this service class will map to UP 3, and thus the Best Effort Access Category (AC_BE). Therefore, a non-default mapping is RECOMMENDED, such that CS4 maps to UP 4, thereby admitting Broadcast Video into the Video Access Category (AC_VI).

4.2.7. Low-Latency Data

The Low-Latency Data service class is recommended for elastic and time-sensitive data applications, often of a transactional nature, where a user is waiting for a response via the network in order to continue with a task at hand. As such, these flows are considered foreground traffic, with delays or drops to such traffic directly impacting user-productivity. [RFC4594] Section 4.7 recommends Low-Latency Data be marked AF2x (that is, AF21, AF22 and AF23, according to the rules defined in [RFC2475]).

By default (as described in Section 2.3), Low-Latency Data will map to UP 2 and thus to the Background Access Category (AC_BK), which is contrary to the intent expressed in [RFC4594].

Mapping Low-Latency Data to UP 3 may allow such to receive a superior level of service via per-UP transmit queues servicing the EDCAF hardware for the Best Effort Access Category (AC_BE), as described in Section 2.2. Therefore it is RECOMMENDED to map Low-Latency Data traffic marked AF2x DSCP to UP 3, thereby admitting it to the Best Effort Access Category (AC_BE).

4.2.8. High-Throughput Data

The High-Throughput Data service class is recommended for elastic applications that require timely packet forwarding of variable rate traffic sources and, more specifically, is configured to provide efficient, yet constrained (when necessary) throughput for TCP longer-lived flows. These flows are typically non-user-interactive. According to [RFC4594] Section 4.8, it can be assumed that this class will consume any available bandwidth and that packets traversing congested links may experience higher queuing delays or packet loss. It is also assumed that this traffic is elastic and responds dynamically to packet loss. [RFC4594] Section 4.8 recommends High-Throughput Data be marked AF1x (that is, AF11, AF12 and AF13, according to the rules defined in [RFC2475]).

By default (as described in Section 2.3), High-Throughput Data will map to UP 1 and thus to the Background Access Category (AC_BK), which is contrary to the intent expressed in [RFC4594].

Unfortunately, there really is no corresponding fit for the High-Throughput Data service class within the constrained 4 Access Category [IEEE.802.11-2016] model. If the High-Throughput Data service class is assigned to the Best Effort Access Category (AC_BE), then it would contend with Low-Latency Data (while [RFC4594] recommends a distinction in servicing between these service classes) as well as with the default service class; alternatively, if it is assigned to the Background Access Category (AC_BK), then it would receive a less-than-best-effort service and contend with Low-Priority Data (as discussed in Section 4.2.10).

As such, since there is no directly corresponding fit for the High-Throughput Data service class within the [IEEE.802.11-2016] model, it is generally RECOMMENDED to map High-Throughput Data to UP 0, thereby admitting it to the Best Effort Access Category (AC_BE).

4.2.9. Standard Service Class

The Standard service class is recommended for traffic that has not been classified into one of the other supported forwarding service classes in the Diffserv network domain. This service class provides the Internet's "best-effort" forwarding behavior. [RFC4594] Section 4.9 states that the "Standard service class MUST use the Default Forwarding (DF) PHB."

The Standard Service Class loosely corresponds to the [IEEE.802.11-2016] Best Effort Access Category (AC_BE) and therefore it is RECOMMENDED to map Standard Service Class traffic marked DF DSCP to UP 0, thereby admitting it to the Best Effort Access Category (AC_BE). This happens to correspond to the default mapping (as described in Section 2.3).

4.2.10. Low-Priority Data

The Low-Priority Data service class serves applications that the user is willing to accept without service assurances. This service class is specified in [RFC3662] and [I-D.ietf-tsvwg-le-phb].

[RFC3662] and [RFC4594] both recommend Low-Priority Data be marked CS1 DSCP.

Note: This marking recommendation may change in the future, as [I-D.ietf-tsvwg-le-phb] defines a Lower Effort (LE) per-hop behavior (PHB) for Low-Priority Data traffic and recommends an additional DSCP for this traffic.

The Low-Priority Data service class loosely corresponds to the [IEEE.802.11-2016] Background Access Category (AC_BK) and therefore

it is RECOMMENDED to map Low-Priority Data traffic marked CS1 DSCP to UP 1, thereby admitting it to the Background Access Category (AC_BK). This happens to correspond to the default mapping (as described in Section 2.3).

4.3. DSCP-to-UP Mapping Recommendations Summary

Figure 1 summarizes the [RFC4594] DSCP marking recommendations mapped to [IEEE.802.11-2016] UP and access categories applied in the downstream direction (i.e. from wired-to-wireless networks).

IETF Diffserv Service Class	PHB	Reference RFC	IEEE 802.11 User Priority	Access Category
Network Control (reserved for future use)	CS7	RFC2474	7 OR 0 See Security Considerations-Sec.8	AC_VO (Voice)
Network Control	CS6	RFC2474	7 OR 0 See Security Considerations-Sec.8	AC_VO (Voice)
Telephony	EF	RFC3246	6	AC_VO (Voice)
VOICE-ADMIT	VA	RFC5865	6	AC_VO (Voice)
Signaling	CS5	RFC2474	5	AC_VI (Video)
Multimedia Conferencing	AF41 AF42 AF43	RFC2597	4	AC_VI (Video)
Real-Time Interactive	CS4	RFC2474	4	AC_VI (Video)
Multimedia Streaming	AF31 AF32 AF33	RFC2597	4	AC_VI (Video)
Broadcast Video	CS3	RFC2474	4	AC_VI (Video)
Low- Latency	AF21 AF22	RFC2597	3	AC_BE (Best Effort)

Data	AF23			
OAM	CS2	RFC2474	0	AC_BE (Best Effort)
High-Throughput Data	AF11	RFC2597	0	AC_BE (Best Effort)
	AF12			
	AF13			
Standard	DF	RFC2474	0	AC_BE (Best Effort)
Low-Priority Data	CS1	RFC3662	1	AC_BK (Background)

Note: All unused codepoints are RECOMMENDED to be mapped to UP 0
(See Security Considerations Section - Section 8)

Figure 1: Summary of Downstream DSCP to IEEE 802.11 UP and AC Mapping Recommendations

5. Upstream Mapping and Marking Recommendations

In the upstream direction (i.e. wireless-to-wired), there are three types of mapping that may be implemented:

- o DSCP-to-UP mapping within the wireless client operating system, and
- o UP-to-DSCP mapping at the wireless access point, or
- o DSCP-Passthrough at the wireless access point (effectively a 1:1 DSCP-to-DSCP mapping)

As an alternative to the latter two options, the network administrator MAY choose to use the wireless-to-wired edge as a Diffserv boundary and explicitly set (or reset) DSCP markings according to administrative policy, thus making the wireless edge a Diffserv policy enforcement point; this approach is RECOMMENDED whenever the APs support the required classification and marking capabilities.

Each of these options will now be considered.

5.1. Upstream DSCP-to-UP Mapping within the Wireless Client Operating System

Some operating systems on wireless client devices utilize a similar default DSCP-to-UP mapping scheme as described in Section 2.3. As such, this can lead to the same conflicts as described in that section, but in the upstream direction.

Therefore, to improve on these default mappings, and to achieve parity and consistency with downstream QoS, it is RECOMMENDED that wireless client operating systems utilize instead the same DSCP-to-UP mapping recommendations presented in Section 4, with the explicit RECOMMENDATION that packets requesting a marking of CS6 or CS7 DSCP SHOULD be mapped to UP 0 (and not to UP 7). Furthermore, in such cases the wireless client operating system SHOULD remark such packets to DSCP 0. This is because CS6 and CS7 DSCP, as well as UP 7 markings, are intended for network control protocols and these SHOULD NOT be sourced from wireless client endpoint devices. This recommendation is detailed in the Security Considerations section (Section 8).

5.2. Upstream UP-to-DSCP Mapping at the Wireless Access Point

UP-to-DSCP mapping generates a DSCP value for the IP packet (either an unencapsulated IP packet or an IP packet encapsulated within a tunneling protocol such as CAPWAP - and destined towards a wireless LAN controller for decapsulation and forwarding) from the Layer 2 [IEEE.802.11-2016] UP marking. This is typically done in the manner described in Section 2.4.

It should be noted that any explicit remarking policy to be performed on such a packet only takes place at the nearest classification and marking policy enforcement point, which may be:

- o At the wireless access point
- o At the wired network switch port
- o At the wireless LAN controller

As such, UP-to-DSCP mapping allows for wireless L2 markings to affect the QoS treatment of a packet over the wired IP network (that is, until the packet reaches the nearest classification and marking policy enforcement point).

It should be further noted that nowhere in the [IEEE.802.11-2016] specifications is there an intent expressed for UP markings to be used to influence QoS treatment over wired IP networks. Furthermore,

[RFC2474], [RFC2475] and [RFC8100] all allow for the host to set DSCP markings for end-to-end QoS treatment over IP networks. Therefore, wireless access points MUST NOT leverage Layer 2 [IEEE.802.11-2016] UP markings as set by wireless hosts and subsequently perform a UP-to-DSCP mapping in the upstream direction. But rather, if wireless host markings are to be leveraged (as per business requirements, technical constraints and administrative policies), then it is RECOMMENDED to pass through the Layer 3 DSCP markings set by these wireless hosts instead, as is discussed in the next section.

5.3. Upstream DSCP-Passthrough at the Wireless Access Point

It is generally NOT RECOMMENDED to pass through DSCP markings from unauthenticated and unauthorized devices, as these are typically considered untrusted sources.

When business requirements and/or technical constraints and/or administrative policies require QoS markings to be passed through at the wireless edge, then it is RECOMMENDED to pass through Layer 3 DSCP markings (over Layer 2 [IEEE.802.11-2016] UP markings) in the upstream direction, with the exception of CS6 and CS7 (as will be discussed further), for the following reasons:

- o [RFC2474], [RFC2475] and [RFC8100] all allow for hosts to set DSCP markings to achieve an end-to-end differentiated service
- o [IEEE.802.11-2016] does not specify that UP markings are to be used to affect QoS treatment over wired IP networks
- o Most present wireless device operating systems generate UP values by the same method as described in Section 2.3 (i.e. by using the 3 MSB of the encapsulated 6-bit DSCP); then, at the access point, these 3-bit markings are converted back into DSCP values, typically in the default manner described in Section 2.4; as such, information is lost in the translation from a 6-bit marking to a 3-bit marking (which is then subsequently translated back to a 6-bit marking); passing through the original (encapsulated) DSCP marking prevents such loss of information
- o A practical implementation benefit is also realized by passing through the DSCP set by wireless client devices, as enabling applications to mark DSCP is much more prevalent and accessible to programmers of applications running on wireless device platforms, vis-a-vis trying to explicitly set UP values, which requires special hooks into the wireless device operating system and/or hardware device drivers, many of which do not support such functionality

CS6 and CS7 are exceptions to this pass through recommendation because wireless hosts SHOULD NOT use them (see Section 5.1) and traffic with those two markings poses a threat to operation of the wired network (see Section 8.2). CS6 and CS7 SHOULD NOT be passed through to the wired network in the upstream direction unless the access point has been specifically configured to do that by a network administrator or operator.

5.4. Upstream DSCP Marking at the Wireless Access Point

An alternative option to mapping is for the administrator to treat the wireless edge as the edge of the Diffserv domain and explicitly set (or reset) DSCP markings in the upstream direction according to administrative policy. This option is RECOMMENDED over mapping, as this typically is the most secure solution, as the network administrator directly enforces the Diffserv policy across the IP network (versus an application developer and/or the wireless endpoint device operating system developer, who may be functioning completely independently of the network administrator).

6. IEEE 802.11 QoS Overview

QoS is enabled on wireless networks by means of the Hybrid Coordination Function (HCF). To give better context to the enhancements in HCF that enable QoS, it may be helpful to begin with a review of the original Distributed Coordination Function (DCF).

6.1. Distributed Coordination Function (DCF)

As has been noted, the Wi-Fi medium is a shared medium, with each station-including the wireless access point-contending for the medium on equal terms. As such, it shares the same challenge as any other shared medium in requiring a mechanism to prevent (or avoid) collisions which can occur when two (or more) stations attempt simultaneous transmission.

The IEEE Ethernet working group solved this challenge by implementing a Carrier Sense Multiple Access/Collision Detection (CSMA/CD) mechanism that could detect collisions over the shared physical cable (as collisions could be detected as reflected energy pulses over the physical wire). Once a collision was detected, then a pre-defined set of rules was invoked that required stations to back off and wait random periods of time before re-attempting transmission. While CSMA/CD improved the usage of Ethernet as a shared medium, it should be noted the ultimate solution to solving Ethernet collisions was the advance of switching technologies, which treated each Ethernet cable as a dedicated collision domain.

However, unlike Ethernet (which uses physical cables), collisions cannot be directly detected over the wireless medium, as RF energy is radiated over the air and colliding bursts are not necessarily reflected back to the transmitting stations. Therefore, a different mechanism is required for this medium.

As such, the IEEE modified the CSMA/CD mechanism to adapt it to wireless networks to provide Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA). The original CSMA/CA mechanism used in IEEE 802.11 was the Distributed Coordination Function. DCF is a timer-based system that leverages three key sets of timers, the slot time, interframe spaces and contention windows.

6.1.1. Slot Time

The slot time is the basic unit of time measure for both DCF and HCF, on which all other timers are based. The slot time duration varies with the different generations of data-rates and performances described by the [IEEE.802.11-2016] standard. For example, the [IEEE.802.11-2016] standard specifies the slot time to be 20 us ([IEEE.802.11-2016] Table 15-5) for legacy implementations (such as IEEE 802.11b, supporting 1, 2, 5.5 and 11 Mbps data rates), while newer implementations (including IEEE 802.11g, 802.11a, 802.11n and 802.11ac, supporting data rates from 6.5 Mbps to over 2 Gbps per spatial stream) define a shorter slot time of 9 us ([IEEE.802.11-2016], Section 17.4.4, Table 17-21).

6.1.2. Interframe Spaces

The time interval between frames that are transmitted over the air is called the Interframe Space (IFS). Several IFS are defined in [IEEE.802.11-2016], with the most relevant to DCF being the Short Interframe Space (SIFS), the DCF Interframe Space (DIFS) and the Extended Interframe Space (EIFS).

The SIFS is the amount of time in microseconds required for a wireless interface to process a received RF signal and its associated [IEEE.802.11-2016] frame and to generate a response frame. Like slot times, the SIFS can vary according to the performance implementation of the [IEEE.802.11-2016] standard. The SIFS for IEEE 802.11a, 802.11n and 802.11ac (in 5 GHz) is 16 us ([IEEE.802.11-2016], Section 17.4.4, Table 17-21).

Additionally, a station must sense the status of the wireless medium before transmitting. If it finds that the medium is continuously idle for the duration of a DIFS, then it is permitted to attempt transmission of a frame (after waiting an additional random backoff period, as will be discussed in the next section). If the channel is

found busy during the DIFS interval, the station must defer its transmission until the medium is found idle for the duration of a DIFS interval. The DIFS is calculated as:

$$\text{DIFS} = \text{SIFS} + (2 * \text{Slot time})$$

However, if all stations waited only a fixed amount of time before attempting transmission then collisions would be frequent. To offset this, each station must wait, not only a fixed amount of time (the DIFS), but also a random amount of time (the random backoff) prior to transmission. The range of the generated random backoff timer is bounded by the Contention Window.

6.1.3. Contention Windows

Contention windows bound the range of the generated random backoff timer that each station must wait (in addition to the DIFS) before attempting transmission. The initial range is set between 0 and the Contention Window minimum value (CWmin), inclusive. The CWmin for DCF (in 5 GHz) is specified as 15 slot times ([IEEE.802.11-2016], Section 17.4.4, Table 17-21).

However, it is possible that two (or more) stations happen to pick the exact same random value within this range. If this happens then a collision may occur. At this point, the stations effectively begin the process again, waiting a DIFS and generate a new random backoff value. However, a key difference is that for this subsequent attempt, the Contention Window approximatively doubles in size (thus exponentially increasing the range of the random value). This process repeats as often as necessary if collisions continue to occur, until the maximum Contention Window size (CWmax) is reached. The CWmax for DCF is specified as 1023 slot times ([IEEE.802.11-2016], Section 17.4.4, Table 17-21).

At this point, transmission attempts may still continue (until some other pre-defined limit is reached), but the Contention Window sizes are fixed at the CWmax value.

Incidentally it may be observed that a significant amount of jitter can be introduced by this contention process for wireless transmission access. For example, the incremental transmission delay of 1023 slot times (CWmax) using 9 us slot times may be as high as 9 ms of jitter per attempt. And, as previously noted, multiple attempts can be made at CWmax.

6.2. Hybrid Coordination Function (HCF)

Therefore, as can be seen from the preceding description of DCF, there is no preferential treatment of one station over another when contending for the shared wireless media; nor is there any preferential treatment of one type of traffic over another during the same contention process. To support the latter requirement, the IEEE enhanced DCF in 2005 to support QoS, specifying HCF in IEEE 802.11, which was integrated into the main IEEE 802.11 standard in 2007.

6.2.1. User Priority (UP)

One of the key changes to the [IEEE.802.11-2016] frame format is the inclusion of a QoS Control field, with 3 bits dedicated for QoS markings. These bits are referred to the User Priority (UP) bits and these support eight distinct marking values: 0-7, inclusive.

While such markings allow for frame differentiation, these alone do not directly affect over-the-air treatment. Rather it is the non-configurable and standard-specified mapping of UP markings to [IEEE.802.11-2016] Access Categories (AC) that generate differentiated treatment over wireless media.

6.2.2. Access Category (AC)

Pairs of UP values are mapped to four defined access categories that correspondingly specify different treatments of frames over the air. These access categories (in order of relative priority from the top down) and their corresponding UP mappings are shown in Figure 2 (adapted from [IEEE.802.11-2016], Section 10.2.4.2, Table 10-1).

User Priority	Access Category	Designative (informative)
7	AC_VO	Voice
6	AC_VO	Voice
5	AC_VI	Video
4	AC_VI	Video
3	AC_BE	Best Effort
0	AC_BE	Best Effort
2	AC_BK	Background
1	AC_BK	Background

Figure 2: IEEE 802.11 Access Categories and User Priority Mappings

The manner in which these four access categories achieve differentiated service over-the-air is primarily by tuning the fixed and random timers that stations have to wait before sending their respective types of traffic, as will be discussed next.

6.2.3. Arbitration Inter-Frame Space (AIFS)

As previously mentioned, each station must wait a fixed amount of time to ensure the medium is idle before attempting transmission. With DCF, the DIFS is constant for all types of traffic. However, with [IEEE.802.11-2016] the fixed amount of time that a station has to wait will depend on the access category and is referred to as an Arbitration Interframe Space (AIFS). AIFS are defined in slot times and the AIFS per access category are shown in Figure 3 (adapted from [IEEE.802.11-2016], Section 9.4.2.29, Table 9-137).

Access Category	Designative (informative)	AIFS (slot times)
AC_VO	Voice	2
AC_VI	Video	2
AC_BE	Best Effort	3
AC_BK	Background	7

Figure 3: Arbitration Interframe Spaces by Access Category

6.2.4. Access Category Contention Windows (CW)

Not only is the fixed amount of time that a station has to wait skewed according to [IEEE.802.11-2016] access category, but so are the relative sizes of the Contention Windows that bound the random backoff timers, as shown in Figure 4 (adapted from [IEEE.802.11-2016], Section 9.4.2.29, Table 9-137).

Access Category	Designative (informative)	CWmin (slot times)	CWmax (slot times)
AC_VO	Voice	3	7
AC_VI	Video	7	15
AC_BE	Best Effort	15	1023
AC_BK	Background	15	1023

Figure 4: Contention Window Sizes by Access Category

When the fixed and randomly generated timers are added together on a per access category basis, then traffic assigned to the Voice Access Category (i.e. traffic marked to UP 6 or 7) will receive a statistically superior service relative to traffic assigned to the Video Access Category (i.e. traffic marked UP 5 and 4), which, in turn, will receive a statistically superior service relative to traffic assigned to the Best Effort Access Category traffic (i.e.

traffic marked UP 3 and 0), which finally will receive a statistically superior service relative to traffic assigned to the Background Access Category traffic (i.e. traffic marked to UP 2 and 1).

6.3. IEEE 802.11u QoS Map Set

IEEE 802.11u [IEEE.802-11u.2011] is an addendum that has now been included within the main [IEEE.802.11-2016] standard, and which includes, among other enhancements, a mechanism by which wireless access points can communicate DSCP to/from UP mappings that have been configured on the wired IP network. Specifically, a QoS Map Set information element (described in [IEEE.802.11-2016] Section 9.4.2.95 and commonly referred to as the QoS Map element) is transmitted from an AP to a wireless endpoint device in an association / re-association Response frame (or within a special QoS Map Configure frame).

The purpose of the QoS Map element is to provide the mapping of higher layer Quality of Service constructs (i.e. DSCP) to User Priorities. One intended effect of receiving such a map is for the wireless endpoint device (that supports this function and is administratively configured to enable it) to perform corresponding DSCP-to-UP mapping within the device (i.e. between applications and the operating system / wireless network interface hardware drivers) to align with what the APs are mapping in the downstream direction, so as to achieve consistent end-to-end QoS in both directions.

The QoS Map element includes two key components:

- 1) each of the eight UP values (0-7) are associated with a range of DSCP values, and
- 2) (up to 21) exceptions from these range-based DSCP to/from UP mapping associations may be optionally and explicitly specified.

In line with the recommendations put forward in this document, the following recommendations apply when the QoS Map element is enabled:

- 1) each of the eight UP values (0-7) are RECOMMENDED to be mapped to DSCP 0 (as a baseline, so as to meet the recommendation made in Section 8.2
- 2) (up to 21) exceptions from this baseline mapping are RECOMMENDED to be made in line with Section 4.3, to correspond to the Diffserv Codepoints that are in use over the IP network.

It is important to note that the QoS Map element is intended to be transmitted from a wireless access point to a non-AP station. As such, the model where this element is used is that of a network where the AP is the edge of the Diffserv domain. Networks where the AP extends the Diffserv domain by connecting other APs and infrastructure devices through the IEEE 802.11 medium are not included in the cases covered by the presence of the QoS Map element, and therefore are not included in the present recommendation.

7. IANA Considerations

This memo asks the IANA for no new parameters.

8. Security Considerations

The recommendations in this document concern widely-deployed wired and wireless network functionality, and for that reason do not present additional security concerns that do not already exist in these networks. In fact, several of the recommendations made in this document serve to protect wired and wireless networks from potential abuse, as is discussed further in this section.

8.1. General QoS Security Recommendations

It may be possible for a wired or wireless device (which could be either a host or a network device) to mark packets (or map packet markings) in a manner that interferes with or degrades existing QoS policies. Such marking or mapping may be done intentionally or unintentionally by developers and/or users and/or administrators of such devices.

To illustrate: A gaming application designed to run on a smart-phone or tablet may request that all its packets be marked DSCP EF and/or UP 6. However, if the traffic from such an application is forwarded without change over a business network, then this could interfere with QoS policies intended to provide priority services for business voice applications.

To mitigate such scenarios it is RECOMMENDED to implement general QoS security measures, including:

- o Setting a traffic conditioning policy reflective of business objectives and policy, such that traffic from authorized users and/or applications and/or endpoints will be accepted by the network; otherwise packet markings will be "bleached" (i.e. remarked to DSCP DF and/or UP 0). Additionally, Section 5.3 made it clear that it is generally NOT RECOMMENDED to pass through DSCP markings from unauthorized and/or unauthenticated devices, as

these are typically considered untrusted sources. This is especially relevant for IoT deployments, where tens-of-billions of devices are being connected to IP networks with little or no security capabilities (making such vulnerable to be utilized as agents for DDoS attacks, the effects of which can be amplified with preferential QoS treatments, should the packet markings of such devices be trusted).

- o Policing EF marked packet flows, as detailed in [RFC2474] Section 7 and [RFC3246] Section 3.

In addition to these general QoS security recommendations, WLAN-specific QoS security recommendations can serve to further mitigate attacks and potential network abuse.

8.2. WLAN QoS Security Recommendations

The wireless LAN presents a unique DoS attack vector, as endpoint devices contend for the shared media on a completely egalitarian basis with the network (as represented by the AP). This means that any wireless client could potentially monopolize the air by sending packets marked to preferred UP values (i.e. UP values 4-7) in the upstream direction. Similarly, airtime could be monopolized if excessive amounts of downstream traffic were marked/mapped to these same preferred UP values. As such, the ability to mark/map to these preferred UP values (of UP 4-7) should be controlled.

If such marking/mapping were not controlled, then, for example, a malicious user could cause WLAN DoS by flooding traffic marked CS7 DSCP downstream. This codepoint would map by default (as described in Section 2.3) to UP 7 and would be assigned to the Voice Access Category (AC_VO). Such a flood could cause Denial-of-Service to not only wireless voice applications, but also to all other traffic classes. Similarly, an uninformed application developer may request all traffic from his/her application to be marked CS7 or CS6, thinking this would achieve in the best overall servicing of their application traffic, while not realizing that such a marking (if honored by the client operating system) could cause not only WLAN DoS, but also IP network instability, as the traffic marked CS7 or CS6 finds its way into queues intended for servicing (relatively low-bandwidth) network control protocols, potentially starving legitimate network control protocols in the process.

Therefore, to mitigate such an attack, it is RECOMMENDED that all packets marked to Diffserv Codepoints not authorized or explicitly provisioned for use over the wireless network by the network administrator be mapped to UP 0; this recommendation applies both at the access point (in the downstream direction) and within the

wireless endpoint device operating system (in the upstream direction).

Such a policy of mapping unused codepoints to UP 0 would also prevent an attack where non-standard codepoints were used to cause WLAN DoS. Consider the case where codepoints are mapped to UP values using a range function (e.g. DSCP values 48-55 all map to UP 6), then an attacker could flood packets marked, for example to DSCP 49, in either the upstream or downstream direction over the WLAN, causing DoS to all other traffic classes in the process.

In the majority of WLAN deployments, the AP represents not only the edge of the Diffserv domain, but also the edge of the network infrastructure itself; that is, only wireless client endpoint devices are downstream from the AP. In such a deployment model, CS6 and CS7 also fall into the category of codepoints that are not in use over the wireless LAN (since only wireless endpoint client devices are downstream from the AP in this model and these devices do not [legitimately] participate in network control protocol exchanges). As such, it is RECOMMENDED that CS6 and CS7 DSCP be mapped to UP 0 in these Wifi-at-the-edge deployment models. Otherwise, it would be easy for a malicious application developer, or even an inadvertently poorly-programmed IoT device, to cause WLAN DoS and even wired IP network instability by flooding traffic marked CS6 DSCP, which would by default (as described in Section 2.3) be mapped to UP 6, causing all other traffic classes on the WLAN to be starved, as well hijacking queues on the wired IP network that are intended for the servicing of routing protocols. To this point, it was also recommended in Section 5.1 that packets requesting a marking of CS6 or CS7 DSCP SHOULD be remarked to DSCP 0 and mapped to UP 0 by the wireless client operating system.

Finally, it should be noted that the recommendations put forward in this document are not intended to address all attack vectors leveraging QoS marking abuse. Mechanisms that may further help mitigate security risks of both wired and wireless networks deploying QoS include strong device- and/or user-authentication, access-control, rate limiting, control-plane policing, encryption and other techniques; however, the implementation recommendations for such mechanisms are beyond the scope of this document to address in detail. Suffice it to say that the security of the devices and networks implementing QoS, including QoS mapping between wired and wireless networks, merits consideration in actual deployments.

9. Acknowledgements

The authors wish to thank David Black, Gorrry Fairhurst, Ruediger Geib, Vincent Roca, Brian Carpenter, David Blake, Cullen Jennings, David Benham and the TSVWG.

The authors also acknowledge a great many inputs, notably from David Kloper, Mark Montanez, Glen Lavers, Michael Fingleton, Sarav Radhakrishnan, Karthik Dakshinamoorthy, Simone Arena, Ranga Marathe, Ramachandra Murthy and many others.

10. References

10.1. Normative References

- [IEEE.802.11-2016]
"Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", IEEE Standard 802.11, 2016, <<https://standards.ieee.org/findstds/standard/802.11-2016.html>>.
- [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>>.
- [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, <<https://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, <<https://www.rfc-editor.org/info/rfc2597>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.

- [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, <<https://www.rfc-editor.org/info/rfc3246>>.
- [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, <<https://www.rfc-editor.org/info/rfc3662>>.
- [RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <<https://www.rfc-editor.org/info/rfc4594>>.
- [RFC5865] Baker, F., Polk, J., and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, DOI 10.17487/RFC5865, May 2010, <<https://www.rfc-editor.org/info/rfc5865>>.
- [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>>.

10.2. Informative References

- [GSMA-IPX_Guidelines]
"Guidelines for IPX Provider networks (Previously Inter-Service Provider IP Backbone Guidelines) Version 11.0", GSMA Official Document, November 2014, <<https://www.gsma.com/newsroom/wp-content/uploads/IR.34-v11.0.pdf>>.
- [I-D.ietf-tsvwg-le-phb]
Bless, R., "A Lower Effort Per-Hop Behavior (LE PHB)", draft-ietf-tsvwg-le-phb-02 (work in progress), June 2017.
- [IEEE.802-11u.2011]
"Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", IEEE Standard 802.11, 2011, <<http://standards.ieee.org/getieee802/download/802.11u-2011.pdf>>.

- [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, <<https://www.rfc-editor.org/info/rfc2475>>.
- [RFC5127] Chan, K., Babiarz, J., and F. Baker, "Aggregation of Diffserv Service Classes", RFC 5127, DOI 10.17487/RFC5127, February 2008, <<https://www.rfc-editor.org/info/rfc5127>>.
- [RFC7561] Kaippallimalil, J., Pazhyannur, R., and P. Yegani, "Mapping Quality of Service (QoS) Procedures of Proxy Mobile IPv6 (PMIPv6) and WLAN", RFC 7561, DOI 10.17487/RFC7561, June 2015, <<https://www.rfc-editor.org/info/rfc7561>>.
- [RFC8100] Geib, R., Ed. and D. Black, "Diffserv-Interconnection Classes and Practice", RFC 8100, DOI 10.17487/RFC8100, March 2017, <<https://www.rfc-editor.org/info/rfc8100>>.

Appendix A. Change Log

Initial Version: July 2015

Authors' Addresses

Tim Szigeti
Cisco Systems
Vancouver, British Columbia V6K 3L4
Canada

Email: szigeti@cisco.com

Jerome Henry
Cisco Systems
Research Triangle Park, North Carolina 27709
USA

Email: jerhenry@cisco.com

Fred Baker
Santa Barbara, California 93117
USA

Email: FredBaker.IETF@gmail.com

Internet Engineering Task Force
Internet-Draft
Updates: 3662,4594 (if approved)
Intended status: Standards Track
Expires: May 18, 2017

R. Bless
Karlsruhe Institute of Technology (KIT)
November 14, 2016

A Lower Effort Per-Hop Behavior (LE PHB)
draft-ietf-tsvwg-le-phb-00

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 May 18, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Applicability	3
1.2. Deployment Considerations	4
1.3. Requirements Language	4
2. PHB Description	4
3. Traffic Conditioning Actions	5
4. Recommended DS Codepoint	5
5. Remarking to other DSCPs/PHBs	5
6. IANA Considerations	6
7. Security Considerations	6
8. References	6
8.1. Normative References	6
8.2. Informative References	6
Appendix A. History of the LE PHB	7
Appendix B. Acknowledgments	7
Author's Address	7

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:

- o For traffic caused by world-wide web search engines while they gather information from web servers.
- o For software updates or dissemination of new releases of operating systems.

- o For backup traffic or non-time critical sychronization or mirroring traffic.
- o For content distribution transfers between caches.
- o For Netnews and other "bulk mail" of the Internet.
- o 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:

- o 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.
- o No parameters or configuration: the LE PHB requires no parameters and no configuration of traffic profiles and so on.
- o 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 76131
Germany

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

Internet Engineering Task Force
Internet-Draft
Obsoletes: 3662 (if approved)
Updates: 4594,8325 (if approved)
Intended status: Standards Track
Expires: September 12, 2019

R. Bless
Karlsruhe Institute of Technology (KIT)
March 11, 2019

A Lower Effort Per-Hop Behavior (LE PHB) for Differentiated Services
draft-ietf-tsvwg-le-phb-10

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. Alternatively, packets forwarded by the LE PHB can be associated with a scavenger service class, i.e., they scavenge otherwise unused resources only. 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. This specification obsoletes RFC 3662 and updates the DSCP recommended in RFC 4594 and RFC 8325 to use the DSCP assigned in this specification.

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. Applicability	3
4. PHB Description	6
5. Traffic Conditioning Actions	7
6. Recommended DS Codepoint	7
7. Deployment Considerations	7
8. Remarking to other DSCPs/PHBs	8
9. Multicast Considerations	9
10. The Update to RFC 4594	10
11. The Update to RFC 8325	12
12. The Update to draft-ietf-tsvwg-rtcweb-qos	12
13. IANA Considerations	14
14. Security Considerations	14
15. References	15
15.1. Normative References	15
15.2. Informative References	15
Appendix A. History of the LE PHB	17
Appendix B. Acknowledgments	18

Appendix C. Change History	18
Appendix D. Note to RFC Editor	21
Author's Address	21

1. Introduction

This document defines a Differentiated Services per-hop behavior [RFC2474] called "Lower Effort" (LE), which is intended for traffic of sufficiently low urgency that all other traffic takes precedence over the LE traffic in consumption of network link bandwidth. Low urgency traffic has a low priority for timely forwarding, 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 packets that ought to consume network resources only when no other traffic is demanding them. In this point of view, packets forwarded by the LE PHB scavenge otherwise unused resources only, which led to the name "scavenger service" in early Internet2 deployments (see Appendix A). Other commonly used names for LE PHB type services are "Lower-than-best-effort" or "Less-than-best-effort". In summary, with the mentioned feature above, the LE PHB has two important properties: it should scavenge residual capacity and it must be preemptable by the default PHB (or other elevated PHBs) in case they need more resources. Consequently, the effect of this type of traffic on all other network traffic is strictly limited ("no harm" property). 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.

2. Requirements Language

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. Applicability

A Lower Effort PHB is applicable for many applications that otherwise use best-effort delivery. More specifically, it is suitable for traffic and services that can tolerate strongly varying throughput

for their data flows, especially periods of very low throughput or even starvation (i.e., long interruptions due to significant or even complete packet loss). Therefore, an application sending an LE marked flow needs to be able to tolerate short or (even very) long interruptions due to the presence of severe congestion conditions during the transmission of the flow. Thus, there ought to be an expectation that packets of the LE PHB could be excessively delayed or dropped when any other traffic is present. It is application-dependent when a lack of progress is considered being a failure (e.g., if a transport connection fails due to timing out, the application may try several times to re-establish the transport connection in order to resume the application session before finally giving up). The LE PHB is suitable for sending traffic of low urgency across a Differentiated Services (DS) domain or DS region.

Just like best-effort traffic, LE traffic SHOULD be congestion controlled (i.e., use a congestion controlled transport or implement an appropriate congestion control method [RFC2914] [RFC8085]). Since LE traffic could be starved completely for a longer period of time, transport protocols or applications (and their related congestion control mechanisms) SHOULD be able to detect and react to such a starvation situation. An appropriate reaction would be to resume the transfer instead of aborting it, i.e., an LE optimized transport ought to use appropriate retry strategies (e.g., exponential back-off with an upper bound) as well as corresponding retry and timeout limits in order to avoid the loss of the connection due to the mentioned starvation periods. While it is desirable to achieve a quick resumption of the transfer as soon as resources become available again, it may be difficult to achieve this in practice. In lack of a transport protocol and congestion control that are adapted to LE, applications can also use existing common transport protocols and implement session resumption by trying to re-establish failed connections. Congestion control is not only useful to let the flows within the LE behavior aggregate adapt to the available bandwidth that may be highly fluctuating, but is also essential if LE traffic is mapped to the default PHB in DS domains that do not support LE. In this case, use of background transport protocols, e.g., similar to LEDBAT [RFC6817], is expedient.

Use of the LE PHB might assist a network operator in moving certain kinds of traffic or users to off-peak times. Furthermore, 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 and packets SHOULD NOT be "downgraded" to the LE PHB instead of being dropped, particularly when the packets are unauthorized traffic. The LE PHB

is expected to have applicability in networks that have at least some unused capacity at certain periods.

The LE PHB allows networks to protect themselves from selected types of traffic as a complement to giving preferential treatment to other selected traffic aggregates. LE ought not to be used for the general case of downgraded traffic, but could 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 in this regard is forwarding of multicast traffic from untrusted sources. Multicast forwarding is currently enabled within domains only for specific sources within a domain, but not for sources from anywhere in the Internet. A major problem is that multicast routing creates traffic sources at (mostly) unpredictable branching points within a domain, potentially leading to congestion and packet loss. In the case of multicast traffic packets from untrusted sources are forwarded as LE traffic, they will not harm traffic from non-LE behavior aggregates. A further related use case is mentioned in [RFC3754]: preliminary forwarding of non-admitted multicast traffic.

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 traffic engineering tool for network administrators. For instance, it can be used to fill protection capacity of transmission links that is otherwise unused. Some network providers keep link utilization below 50% to ensure that all traffic is forwarded without loss after rerouting caused by a link failure (cf. Section 6 of [RFC3439]). 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. Since LE traffic may be starved for longer periods of time it is probably less suitable for real-time and interactive applications.

Example uses for the LE PHB:

- o For traffic caused by world-wide web search engines while they gather information from web servers.
- o For software updates or dissemination of new releases of operating systems.
- o For reporting errors or telemetry data from operating systems or applications.

- o For backup traffic or non-time critical synchronization or mirroring traffic.
- o For content distribution transfers between caches.
- o For preloading or prefetching objects from web sites.
- o For network news and other "bulk mail" of the Internet.
- o For "downgraded" traffic from some other PHB when this does not violate the operational objectives of the other PHB.
- o For multicast traffic from untrusted (e.g., non-local) sources.

4. PHB Description

The LE PHB is defined in relation to the default PHB (best-effort). A packet forwarded with the LE PHB SHOULD have lower precedence than packets forwarded with the default PHB, i.e., in the case of congestion, LE marked traffic SHOULD be dropped prior to dropping any default PHB traffic. Ideally, LE packets would be forwarded only when no packet with any other PHB is awaiting transmission. This means that in case of link resource contention LE traffic can be starved completely, which may not be always desired by the network operator's policy. The used scheduler to implement the LE PHB may reflect this policy accordingly.

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, could 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.

If a dedicated LE queue is not available, an active queue management mechanism within a common BE/LE queue could also be used. This could drop all arriving LE packets as soon as certain queue length or sojourn time thresholds are exceeded.

Since congestion control is also useful within the LE traffic class, Explicit Congestion Notification (ECN) [RFC3168] SHOULD be used for LE packets, too. More specifically, an LE implementation SHOULD also apply CE marking for ECT marked packets and transport protocols used for LE SHOULD support and employ ECN. For more information on the benefits of using ECN see [RFC8087].

5. Traffic Conditioning Actions

If possible, packets SHOULD 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. Thus, any policing such as limiting the rate of LE traffic is not necessary at the DS boundary.

As for most other PHBs an initial classification and marking can be also performed at the first DS boundary node according to the DS domain's own policies (e.g., as protection measure against untrusted sources). However, non-LE traffic (e.g., BE traffic) SHOULD NOT be remarked to LE. Remarketing traffic from another PHB results in that traffic being "downgraded". This changes the way the network treats this traffic and it is important not to violate the operational objectives of the original PHB. See also remarks with respect to downgrading in Section 3 and Section 8.

6. Recommended DS Codepoint

The RECOMMENDED codepoint for the LE PHB is '000001'.

Earlier specifications [RFC4594] recommended to use CS1 as codepoint (as mentioned in [RFC3662]). This is problematic since it may cause a priority inversion in Diffserv domains that treat CS1 as originally proposed in [RFC2474], resulting in forwarding LE packets with higher precedence than BE packets. Existing implementations SHOULD transition to use the unambiguous LE codepoint '000001' whenever possible.

This particular codepoint was chosen due to measurements on the currently observable DSCP remarking behavior in the Internet [ietf99-secchi]. Since some network domains set the former IP precedence bits to zero, it is possible that some other standardized DSCPs get mapped to the LE PHB DSCP if it were taken from the DSCP standards action pool 1 (xxxxx0).

7. Deployment Considerations

In order to enable LE support, DS nodes typically only need

- o A BA classifier (Behavior Aggregate classifier, see [RFC2475]) that classifies packets according to the LE DSCP
- o A dedicated LE queue
- o A suitable scheduling discipline, e.g., simple priority queueing

Alternatively, implementations could use active queue management mechanisms instead of a dedicated LE queue, e.g., dropping all arriving LE packets when certain queue length or sojourn time thresholds are exceeded.

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

- o No harm to other traffic: since the LE PHB has the lowest forwarding priority it does not consume resources from other PHBs. Deployment across different provider domains with LE support causes no trust issues or attack vectors to existing (non LE) traffic. Thus, providers can trust LE markings from end-systems, i.e., there is no need to police or remark incoming LE traffic.
- o No PHB parameters or configuration of traffic profiles: the LE PHB itself possesses no parameters that need to be set or configured. Similarly, since LE traffic requires no admission or policing, it is not necessary to configure traffic profiles.
- o No traffic conditioning mechanisms: the LE PHB requires no traffic meters, droppers, or shapers. See also Section 5 for further discussion.

Operators of DS domains that cannot or do not want to implement the LE PHB (e.g., because there is no separate LE queue available in the corresponding nodes) SHOULD NOT drop packets marked with the LE DSCP. They SHOULD map packets with this DSCP to the default PHB and SHOULD preserve the LE DSCP marking. DS domains operators that do not implement the LE PHB should be aware that they violate the "no harm" property of LE. See also Section 8 for further discussion of forwarding LE traffic with the default PHB instead.

8. 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 (no harm property). In the case that a DS domain does not support the LE PHB, its nodes SHOULD treat LE marked packets with the default PHB instead (by mapping the LE DSCP to the default PHB), but they SHOULD do so without remarking to DSCP '000000'. The reason for this is that later traversed DS domains may then have still the possibility to treat such packets according to the LE PHB.

Operators of DS domains that forward LE traffic within the BE aggregate need to be aware of the implications, i.e., induced congestion situations and quality-of-service degradation of the

original BE traffic. In this case, the LE property of not harming other traffic is no longer fulfilled. To limit the impact in such cases, traffic policing of the LE aggregate MAY be used.

In the case that LE marked packets are effectively carried within the default PHB (i.e., forwarded as best-effort traffic) they get a better forwarding treatment than expected. For some applications and services, it is favorable if the transmission is finished earlier than expected. However, in some cases it may be against the original intention of the LE PHB user to strictly send the traffic only if otherwise unused resources are available. In the case that LE traffic is mapped to the default PHB, LE traffic may compete with BE traffic for the same resources and thus adversely affect the original BE aggregate. Applications that want to ensure the lower precedence compared to BE traffic even in such cases SHOULD use additionally a corresponding Lower-than-Best-Effort transport protocol [RFC6297], e.g., LEDBAT [RFC6817].

A DS domain that still uses DSCP CS1 for marking LE traffic (including Low Priority-Data as defined in [RFC4594] or the old definition in [RFC3662]) SHOULD remark traffic to the LE DSCP '000001' at the egress to the next DS domain. This increases the probability that the DSCP is preserved end-to-end, whereas a CS1 marked packet may be remarked by the default DSCP if the next domain is applying Diffserv-Interconnection [RFC8100].

9. Multicast Considerations

Basically, the multicast considerations in [RFC3754] apply. However, using the Lower Effort PHB for multicast requires paying special attention to the way how packets get replicated inside routers. Due to multicast packet replication, resource contention may actually occur even before a packet is forwarded to its output port and in the worst case, these forwarding resources are missing for higher prioritized multicast or even unicast packets.

Several forward error correction coding schemes such as fountain codes (e.g., [RFC5053]) allow reliable data delivery even in environments with a potential high amount of packet loss in transmission. When used for example over satellite links or other broadcast media, this means that receivers that lose 80% of packets in transmission simply need 5 times as long to receive the complete data than those receivers experiencing no loss (without any receiver feedback required).

Superficially viewed, it may sound very attractive to use IP multicast with the LE PHB to build this type of opportunistic reliable distribution in IP networks, but it can only be usefully

deployed with routers that do not experience forwarding/replication resource starvation when a large amount of packets (virtually) need to be replicated to links where the LE queue is full.

Thus, packet replication of LE marked packets should consider the situation at the respective output links: it is a waste of internal forwarding resources if a packet is replicated to output links that have no resources left for LE forwarding. In those cases a packet would have been replicated just to be dropped immediately after finding a filled LE queue at the respective output port. Such behavior could be avoided for example by using a conditional internal packet replication: a packet would then only be replicated in case the output link is not fully used. This conditional replication, however, is probably not widely implemented.

While the resource contention problem caused by multicast packet replication is also true for other Diffserv PHBs, LE forwarding is special, because often it is assumed that LE packets only get forwarded in case of available resources at the output ports. The previously mentioned redundancy data traffic could nicely use the varying available residual bandwidth being utilized by LE PHB, but only if the specific requirements stated above for conditional replication in the internal implementation of the network devices are considered.

10. The Update to RFC 4594

[RFC4594] recommended to use CS1 as codepoint in section 4.10, whereas CS1 was defined in [RFC2474] to have a higher precedence than CS0, i.e., the default PHB. Consequently, Diffserv domains implementing CS1 according to [RFC2474] will cause a priority inversion for LE packets that contradicts with the original purpose of LE. Therefore, every occurrence of the CS1 DSCP is replaced by the LE DSCP.

Changes:

- o This update to RFC 4594 removes the following entry from figure 3:

Low-Priority Data	CS1	001000	Any flow that has no BW assurance
----------------------	-----	--------	--------------------------------------

and replaces this by the following entry:

Low-Priority Data	LE	000001	Any flow that has no BW assurance
----------------------	----	--------	--------------------------------------

- o This update to RFC 4594 extends the Notes text below figure 3 that currently states "Notes for Figure 3: Default Forwarding (DF) and Class Selector 0 (CS0) provide equivalent behavior and use the same DS codepoint, '000000'." to state "Notes for Figure 3: Default Forwarding (DF) and Class Selector 0 (CS0) provide equivalent behavior and use the same DS codepoint, '000000'. The prior recommendation to use the CS1 DSCP for Low-Priority Data has been replaced by the current recommendation to use the LE DSCP, '000001'."
- o This update to RFC 4594 removes the following entry from figure 4:

Low-Priority Data	CS1	Not applicable	RFC3662	Rate	Yes
----------------------	-----	----------------	---------	------	-----

and replaces this by the following entry:

Low-Priority Data	LE	Not applicable	RFCXXXX	Rate	Yes
----------------------	----	----------------	---------	------	-----

- o Section 2.3 of [RFC4594] specifies: "In network segments that use IP precedence marking, only one of the two service classes can be supported, High-Throughput Data or Low-Priority Data. We RECOMMEND that the DSCP value(s) of the unsupported service class be changed to 000xx1 on ingress and changed back to original value(s) on egress of the network segment that uses precedence marking. For example, if Low-Priority Data is mapped to Standard service class, then 000001 DSCP marking MAY be used to distinguish it from Standard marked packets on egress." This document removes this recommendation, because by using the herein defined LE DSCP such remarking is not necessary. So even if Low-Priority Data is unsupported (i.e., mapped to the default PHB) the LE DSCP should be kept across the domain as RECOMMENDED in Section 8. That removed text is replaced by: "In network segments that use IP Precedence marking, the Low-Priority Data service class receives the same Diffserv QoS as the Standard service class when the LE DSCP is used for Low-Priority Data traffic. This is acceptable behavior for the Low-Priority Data service class, although it is not the preferred behavior."

- o This document removes the following line of RFC 4594, Section 4.10: "The RECOMMENDED DSCP marking is CS1 (Class Selector 1)." and replaces this with the following text: "The RECOMMENDED DSCP marking is LE (Lower Effort), which replaces the prior recommendation for CS1 (Class Selector 1) marking."

11. The Update to RFC 8325

Section 4.2.10 of RFC 8325 [RFC8325] specifies "[RFC3662] and [RFC4594] both recommend Low-Priority Data be marked CS1 DSCP." which is updated to "[RFC3662] recommends that Low-Priority Data be marked CS1 DSCP. [RFC4594] as updated by [RFCXXXX] recommends Low-Priority Data be marked LE DSCP."

This document removes the following paragraph of RFC 8325, Section 4.2.10 because this document makes the anticipated change: "Note: This marking recommendation may change in the future, as [LE-PHB] defines a Lower Effort (LE) PHB for Low-Priority Data traffic and recommends an additional DSCP for this traffic."

Section 4.2.10 of RFC 8325 [RFC8325] specifies "therefore, it is RECOMMENDED to map Low-Priority Data traffic marked CS1 DSCP to UP 1" which is updated to "therefore, it is RECOMMENDED to map Low-Priority Data traffic marked with LE DSCP or legacy CS1 DSCP to UP 1"

This update to RFC 8325 replaces the following entry from figure 1:

Low-Priority Data	CS1	RFC 3662	1	AC_BK (Background)
-------------------	-----	----------	---	--------------------

by the following entries:

Low-Priority Data	LE	RFCXXXX	1	AC_BK (Background)
Low-Priority Data (legacy)	CS1	RFC 3662	1	AC_BK (Background)

12. The Update to draft-ietf-tsvwg-rtcweb-qos

Section 5 of [I-D.ietf-tsvwg-rtcweb-qos] describes the Recommended DSCP Values for WebRTC Applications

This update to [I-D.ietf-tsvwg-rtcweb-qos] replaces all occurrences of CS1 with LE in Table 1:

Flow Type	Very Low	Low	Medium	High
Audio	LE (1)	DF (0)	EF (46)	EF (46)
Interactive Video with or without Audio	LE (1)	DF (0)	AF42, AF43 (36, 38)	AF41, AF42 (34, 36)
Non-Interactive Video with or without Audio	LE (1)	DF (0)	AF32, AF33 (28, 30)	AF31, AF32 (26, 28)
Data	LE (1)	DF (0)	AF11	AF21

and updates the following paragraph:

"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."

as follows:

"The above table assumes that packets marked with LE are treated as lower effort (i.e., "less than best effort"), such as the LE behavior described in [RFCXXXX]. However, the treatment of LE is implementation dependent. If an implementation treats LE 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 LE to be treated the same as DF, so applications and browsers using LE cannot assume that LE will be treated differently than DF [RFC7657]. During development of this document, the CS1 DSCP was recommended for "very low" application

priority traffic; implementations that followed that recommendation SHOULD be updated to use the LE DSCP instead of the CS1 DSCP."

13. IANA Considerations

This document assigns the Differentiated Services Field Codepoint (DSCP) '000001' from the Differentiated Services Field Codepoints (DSCP) registry (<https://www.iana.org/assignments/dscp-registry/dscp-registry.xhtml>) (Pool 3, Codepoint Space xxxx01, Standards Action) to the LE PHB. This document suggests to use a DSCP from Pool 3 in order to avoid problems for other PHB marked flows to become accidentally remarked as LE PHB, e.g., due to partial DSCP bleaching. See [RFC8436] for re-classifying Pool 3 for Standards Action.

IANA is requested to update the registry as follows:

- o Name: LE
- o Value (Binary): 000001
- o Value (Decimal): 1
- o Reference: [RFC number of this memo]

14. Security Considerations

There are no specific security exposures for this PHB. Since it defines a new class of low forwarding priority, remarking other traffic as LE traffic may lead to quality-of-service degradation of such traffic. Thus, any attacker that is able to modify the DSCP of a packet to LE may carry out a downgrade attack. See the general security considerations in [RFC2474] and [RFC2475].

With respect to privacy, an attacker could use the information from the DSCP to infer that the transferred (probably even encrypted) content is considered of low priority or low urgency by a user, in case the DSCP was set on the user's request. On the one hand, this disclosed information is useful only if correlation with metadata (such as the user's IP address) and/or other flows reveal user identity. On the other hand, it might help an observer (e.g., a state level actor) who is interested in learning about the user's behavior from observed traffic: LE marked background traffic (such as software downloads, operating system updates, or telemetry data) may be less interesting for surveillance than general web traffic. Therefore, the LE marking may help the observer to focus on potentially more interesting traffic (however, the user may exploit this particular assumption and deliberately hide interesting traffic in the LE aggregate). Apart from such considerations, the impact of

disclosed information by the LE DSCP is likely negligible in most cases given the numerous traffic analysis possibilities and general privacy threats (e.g., see [RFC6973]).

15. References

15.1. Normative References

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

15.2. Informative References

- [carlberg-lbe-2001] Carlberg, K., Gevros, P., and J. Crowcroft, "Lower than best effort: a design and implementation", SIGCOMM Computer Communications Review Volume 31, Issue 2 supplement, April 2001, <<https://doi.org/10.1145/844193.844208>>.
- [chown-lbe-2003] Chown, T., Ferrari, T., Leinen, S., Sabatino, R., Simar, N., and S. Venaas, "Less than Best Effort: Application Scenarios and Experimental Results", In Proceedings of the Second International Workshop on Quality of Service in Multiservice IP Networks (QoS-IP 2003), Lecture Notes in Computer Science, vol 2601. Springer, Berlin, Heidelberg Pages 131-144, February 2003, <https://doi.org/10.1007/3-540-36480-3_10>.

- [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>>.
- [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-18 (work in progress), August 2016.
- [ietf99-secchi]
Secchi, R., Venne, A., and A. Custura, "Measurements concerning the DSCP for a LE PHB", Presentation held at 99th IETF Meeting, TSVWG, Prague, July 2017, <<https://datatracker.ietf.org/meeting/99/materials/slides-99-tsvwg-sessb-31measurements-concerning-the-dscp-for-a-le-phb-00>>.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<https://www.rfc-editor.org/info/rfc2914>>.
- [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>>.
- [RFC3439] Bush, R. and D. Meyer, "Some Internet Architectural Guidelines and Philosophy", RFC 3439, DOI 10.17487/RFC3439, December 2002, <<https://www.rfc-editor.org/info/rfc3439>>.
- [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>>.

- [RFC5053] Luby, M., Shokrollahi, A., Watson, M., and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery", RFC 5053, DOI 10.17487/RFC5053, October 2007, <<https://www.rfc-editor.org/info/rfc5053>>.
- [RFC6297] Welzl, M. and D. Ros, "A Survey of Lower-than-Best-Effort Transport Protocols", RFC 6297, DOI 10.17487/RFC6297, June 2011, <<http://www.rfc-editor.org/info/rfc6297>>.
- [RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <<http://www.rfc-editor.org/info/rfc6817>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <<https://www.rfc-editor.org/info/rfc8087>>.
- [RFC8100] Geib, R., Ed. and D. Black, "Diffserv-Interconnection Classes and Practice", RFC 8100, DOI 10.17487/RFC8100, March 2017, <<http://www.rfc-editor.org/info/rfc8100>>.
- [RFC8325] Szigeti, T., Henry, J., and F. Baker, "Mapping Diffserv to IEEE 802.11", RFC 8325, DOI 10.17487/RFC8325, February 2018, <<https://www.rfc-editor.org/info/rfc8325>>.
- [RFC8436] Fairhurst, G., "Update to IANA Registration Procedures for Pool 3 Values in the Differentiated Services Field Codepoints (DSCP) Registry", RFC 8436, DOI 10.17487/RFC8436, August 2018, <<https://www.rfc-editor.org/info/rfc8436>>.

Appendix A. History of the LE PHB

A first version of this PHB was suggested by Roland Bless and Klaus Wehrle in September 1999 [draft-bless-diffserv-lbe-phb-00], named "A Lower Than Best-Effort Per-Hop Behavior". 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].

There are several other names in use for this type of PHB or associated service classes. Well-known is the QBone Scavenger Service (QBSS) that was proposed in March 2001 within the Internet2 QoS Working Group. Alternative names are "Lower-than-best-effort" [carlberg-lbe-2001] or "Less-than-best-effort" [chown-lbe-2003].

Appendix B. Acknowledgments

Since text is partially borrowed from earlier Internet-Drafts and RFCs the co-authors of previous specifications are acknowledged here: Kathie Nichols and Klaus Wehrle. David Black, Olivier Bonaventure, Spencer Dawkins, Toerless Eckert, Gorrry Fairhurst, Ruediger Geib, and Kyle Rose provided helpful comments and (partially also text) suggestions.

Appendix C. Change History

This section briefly lists changes between Internet-Draft versions for convenience.

Changes in Version 10: (incorporated comments from IESG discussion as follows)

- o Appended "for Differentiated Services" to the title as suggested by Alexey.
- o Addressed Deborah Brungard's discuss: changed phrase to "However, non-LE traffic (e.g., BE traffic) SHOULD NOT be remarked to LE." with additional explanation as suggested by Gorrry.
- o Fixed the sentence "An LE PHB SHOULD NOT be used for a customer's "normal Internet" traffic nor should packets be "downgraded" to the LE PHB instead of being dropped, particularly when the packets are unauthorized traffic." according to Alice's and Mirja's comments.
- o Made reference to RFC8174 normative.
- o Added hint for the RFC editor to apply changes from section Section 12 and to delete it afterwards.

- o Incorporated Mirja's and Benjamin's suggestions.
- o Editorial suggested by Gorrry: In case => In the case that

Changes in Version 09:

- o Incorporated comments from IETF Last Call:
 - * from Olivier Bonaventure: added a bit of text for session resumption and congestion control aspects as well as ECN usage.
 - * from Kyle Rose: Revised privacy considerations text in Security Considerations Section

Changes in Version 08:

- o revised two sentences as suggested by Spencer Dawkins

Changes in Version 07:

- o revised some text for clarification according to comments from Spencer Dawkins

Changes in Version 06:

- o added Multicast Considerations section with input from Toerless Eckert
- o incorporated suggestions by David Black with respect to better reflect legacy CS1 handling

Changes in Version 05:

- o added scavenger service class into abstract
- o added some more history
- o added reference for "Myth of Over-Provisioning" in RFC3439 and references to presentations w.r.t. codepoint choices
- o added text to update draft-ietf-tsvwg-rtcweb-qos
- o revised text on congestion control in case of remarking to BE
- o added reference to DSCP measurement talk @IETF99
- o small typo fixes

Changes in Version 04:

- o Several editorial changes according to review from Gorrry Fairhurst
- o Changed the section structure a bit (moved subsections 1.1 and 1.2 into own sections 3 and 7 respectively)
- o updated section 2 on requirements language
- o added updates to RFC 8325
- o tried to be more explicit what changes are required to RFCs 4594 and 8325

Changes in Version 03:

- o Changed recommended codepoint to 000001
- o Added text to explain the reasons for the DSCP choice
- o Removed LE-min,LE-strict discussion
- o Added one more potential use case: reporting errors or telemetry data from OSs
- o Added privacy considerations to the security section (not worth an own section I think)
- o Changed IANA considerations section

Changes in Version 02:

- o Applied many editorial suggestions from David Black
- o Added Multicast traffic use case
- o Clarified what is required for deployment in section 1.2 (Deployment Considerations)
- o Added text about implementations using AQMs and ECN usage
- o Updated IANA section according to David Black's suggestions
- o Revised text in the security section
- o Changed copyright Notice to pre5378Trust200902

Changes in Version 01:

- o Now obsoletes RFC 3662.
- o Tried to be more precise in section 1.1 (Applicability) according to R. Geib's suggestions, so rephrased several paragraphs. Added text about congestion control
- o Change section 2 (PHB Description) according to R. Geib's suggestions.
- o Added RFC 2119 language to several sentences.
- o Detailed the description of remarking implications and recommendations in Section 8.
- o Added Section 10 to explicitly list changes with respect to RFC 4594, because this document will update it.

Appendix D. Note to RFC Editor

This section lists actions for the RFC editor during final formatting.

- o Apply the suggested changes of section Section 12 and add a normative reference in draft-ietf-tsvwg-rtcweb-qos to this RFC.
- o Delete Section 12.
- o Please replace the occurrences of RFCXXXX in Section 10 and Section 11 with the assigned RFC number for this document.
- o Delete Appendix C.
- o Delete this section.

Author's Address

Roland Bless
Karlsruhe Institute of Technology (KIT)
Kaiserstr. 12
Karlsruhe 76131
Germany

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 April 2022

R. R. Stewart
Netflix, Inc.
M. Tüxen
I. Rüngeler
Münster Univ. of Appl. Sciences
25 October 2021

Stream Control Transmission Protocol (SCTP) Network Address Translation
Support
draft-ietf-tsvwg-natsupp-23

Abstract

The Stream Control Transmission Protocol (SCTP) provides a reliable communications channel between two end-hosts in many ways similar to the Transmission Control Protocol (TCP). With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT functions for TCP that allows multiple hosts to reside behind a NAT function and yet share a single IPv4 address, even when two hosts (behind a NAT function) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation (NAPT).

This document describes the protocol extensions needed for the SCTP endpoints and the mechanisms for NAT functions necessary to provide similar features of NAPT in the single point and multipoint traversal scenario.

Finally, a YANG module for SCTP NAT is defined.

Status of This Memo

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 28 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions	5
3. Terminology	5
4. Motivation and Overview	6
4.1. SCTP NAT Traversal Scenarios	6
4.1.1. Single Point Traversal	7
4.1.2. Multipoint Traversal	7
4.2. Limitations of Classical NAPT for SCTP	8
4.3. The SCTP-Specific Variant of NAT	8
5. Data Formats	13
5.1. Modified Chunks	13
5.1.1. Extended ABORT Chunk	13
5.1.2. Extended ERROR Chunk	14
5.2. New Error Causes	14
5.2.1. VTag and Port Number Collision Error Cause	14
5.2.2. Missing State Error Cause	15
5.2.3. Port Number Collision Error Cause	15
5.3. New Parameters	16
5.3.1. Disable Restart Parameter	16
5.3.2. VTags Parameter	17
6. Procedures for SCTP Endpoints and NAT Functions	18
6.1. Association Setup Considerations for Endpoints	19
6.2. Handling of Internal Port Number and Verification Tag Collisions	19
6.2.1. NAT Function Considerations	19
6.2.2. Endpoint Considerations	20
6.3. Handling of Internal Port Number Collisions	20
6.3.1. NAT Function Considerations	20
6.3.2. Endpoint Considerations	21
6.4. Handling of Missing State	21
6.4.1. NAT Function Considerations	22
6.4.2. Endpoint Considerations	22

6.5.	Handling of Fragmented SCTP Packets by NAT Functions . .	24
6.6.	Multi Point Traversal Considerations for Endpoints . . .	24
7.	SCTP NAT YANG Module	24
7.1.	Tree Structure	24
7.2.	YANG Module	25
8.	Various Examples of NAT Traversals	27
8.1.	Single-homed Client to Single-homed Server	28
8.2.	Single-homed Client to Multi-homed Server	30
8.3.	Multihomed Client and Server	32
8.4.	NAT Function Loses Its State	35
8.5.	Peer-to-Peer Communications	37
9.	Socket API Considerations	42
9.1.	Get or Set the NAT Friendliness (SCTP_NAT_FRIENDLY) . . .	43
10.	IANA Considerations	43
10.1.	New Chunk Flags for Two Existing Chunk Types	43
10.2.	Three New Error Causes	45
10.3.	Two New Chunk Parameter Types	46
10.4.	One New URI	46
10.5.	One New YANG Module	46
11.	Security Considerations	46
12.	Normative References	47
13.	Informative References	48
	Acknowledgments	51
	Authors' Addresses	51

1. Introduction

Stream Control Transmission Protocol (SCTP) [RFC4960] provides a reliable communications channel between two end-hosts in many ways similar to TCP [RFC0793]. With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT functions for TCP that allows multiple hosts to reside behind a NAT function using private-use addresses (see [RFC6890]) and yet share a single IPv4 address, even when two hosts (behind a NAT function) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation (NAPT). Please note that this document focuses on the case where the NAT function maps a single or multiple internal addresses to a single external address and vice versa.

To date, specialized code for SCTP has not yet been added to most NAT functions so that only a translation of IP addresses is supported. The end result of this is that only one SCTP-capable host can successfully operate behind such a NAT function and this host can only be single-homed. The only alternative for supporting legacy NAT functions is to use UDP encapsulation as specified in [RFC6951].

The NAT function in the document refers to NAPT functions described in Section 2.2 of [RFC3022], NAT64 [RFC6146], or DS-Lite AFTR [RFC6333].

This document specifies procedures allowing a NAT function to support SCTP by providing similar features to those provided by a NAPT for TCP (see [RFC5382] and [RFC7857]), UDP (see [RFC4787] and [RFC7857]), and ICMP (see [RFC5508] and [RFC7857]). This document also specifies a set of data formats for SCTP packets and a set of SCTP endpoint procedures to support NAT traversal. An SCTP implementation supporting these procedures can assure that in both single-homed and multi-homed cases a NAT function will maintain the appropriate state without the NAT function needing to change port numbers.

It is possible and desirable to make these changes for a number of reasons:

- * It is desirable for SCTP internal end-hosts on multiple platforms to be able to share a NAT function's external IP address in the same way that a TCP session can use a NAT function.
- * If a NAT function does not need to change any data within an SCTP packet, it will reduce the processing burden of NAT'ing SCTP by not needing to execute the CRC32c checksum used by SCTP.
- * Not having to touch the IP payload makes the processing of ICMP messages by NAT functions easier.

An SCTP-aware NAT function will need to follow these procedures for generating appropriate SCTP packet formats.

When considering SCTP-aware NAT it is possible to have multiple levels of support. At each level, the Internal Host, Remote Host, and NAT function does or does not support the procedures described in this document. The following table illustrates the results of the various combinations of support and if communications can occur between two endpoints.

Internal Host	NAT Function	Remote Host	Communication
Support	Support	Support	Yes
Support	Support	No Support	Limited
Support	No Support	Support	None
Support	No Support	No Support	None
No Support	Support	Support	Limited
No Support	Support	No Support	Limited
No Support	No Support	Support	None
No Support	No Support	No Support	None

Table 1: Communication possibilities

From the table it can be seen that no communication can occur when a NAT function does not support SCTP-aware NAT. This assumes that the NAT function does not handle SCTP packets at all and all SCTP packets sent from behind a NAT function are discarded by the NAT function. In some cases, where the NAT function supports SCTP-aware NAT, but one of the two hosts does not support the feature, communication can possibly occur in a limited way. For example, only one host can have a connection when a collision case occurs.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

This document uses the following terms, which are depicted in Figure 1. Familiarity with the terminology used in [RFC4960] and [RFC5061] is assumed.

Internal-Address (Int-Addr)

An internal address that is known to the internal host.

Internal-Port (Int-Port)

The port number that is in use by the host holding the Internal-Address.

Internal-VTag (Int-VTag)

The SCTP Verification Tag (VTag) (see Section 3.1 of [RFC4960]) that the internal host has chosen for an association. The VTag is a unique 32-bit tag that accompanies any incoming SCTP packet for this association to the Internal-Address.

Remote-Address (Rem-Addr)

The address that an internal host is attempting to contact.

Remote-Port (Rem-Port)

The port number used by the host holding the Remote-Address.

Remote-VTag (Rem-VTag)

The Verification Tag (VTag) (see Section 3.1 of [RFC4960]) that the host holding the Remote-Address has chosen for an association. The VTag is a unique 32-bit tag that accompanies any outgoing SCTP packet for this association to the Remote-Address.

External-Address (Ext-Addr)

An external address assigned to the NAT function, that it uses as a source address when sending packets towards a Remote-Address.

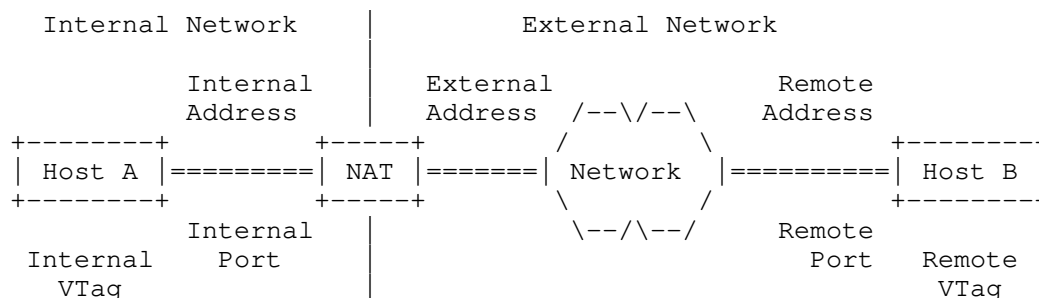


Figure 1: Basic Network Setup

4. Motivation and Overview

4.1. SCTP NAT Traversal Scenarios

This section defines the notion of single and multipoint NAT traversal.

4.1.1. Single Point Traversal

In this case, all packets in the SCTP association go through a single NAT function, as shown in Figure 2.

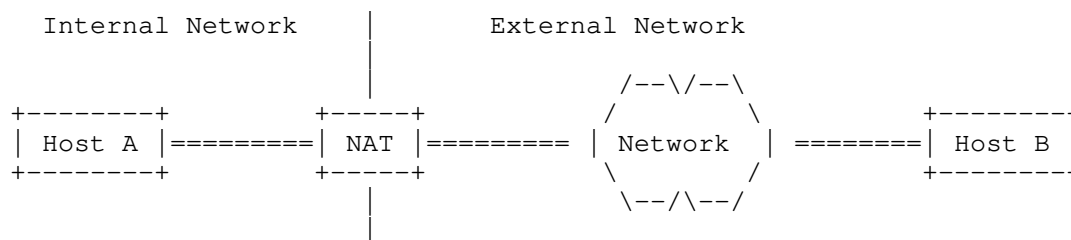


Figure 2: Single NAT Function Scenario

A variation of this case is shown in Figure 3, i.e., multiple NAT functions in the forwarding path between two endpoints.

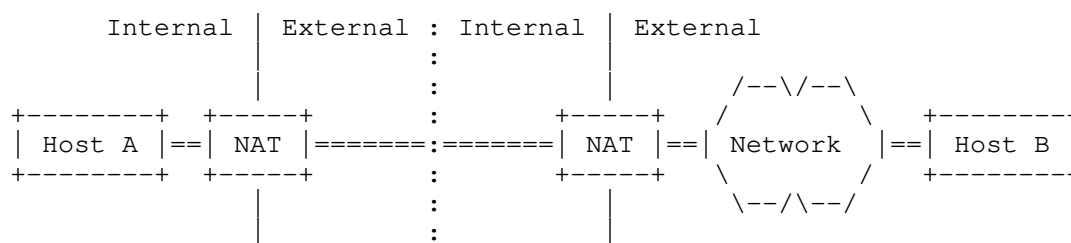


Figure 3: Serial NAT Functions Scenario

Although one of the main benefits of SCTP multi-homing is redundant paths, in the single point traversal scenario the NAT function represents a single point of failure in the path of the SCTP multi-homed association. However, the rest of the path can still benefit from path diversity provided by SCTP multi-homing.

The two SCTP endpoints in this case can be either single-homed or multi-homed. However, the important thing is that the NAT function in this case sees all the packets of the SCTP association.

4.1.2. Multipoint Traversal

This case involves multiple NAT functions and each NAT function only sees some of the packets in the SCTP association. An example is shown in Figure 4.

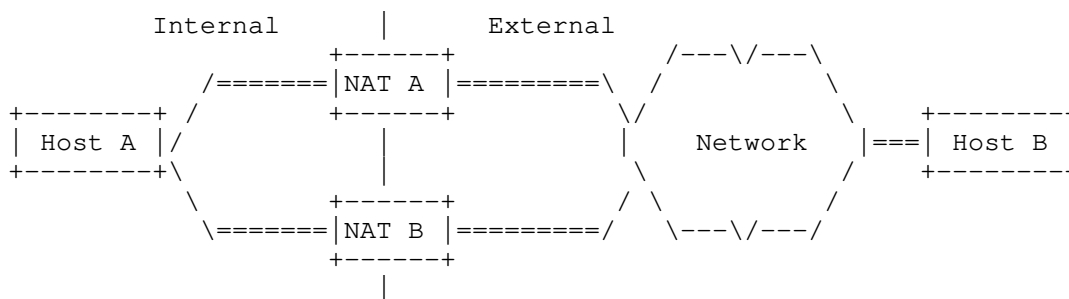


Figure 4: Parallel NAT Functions Scenario

This case does not apply to a single-homed SCTP association (i.e., both endpoints in the association use only one IP address). The advantage here is that the existence of multiple NAT traversal points can preserve the path diversity of a multi-homed association for the entire path. This in turn can improve the robustness of the communication.

4.2. Limitations of Classical NAPT for SCTP

Using classical NAPT possibly results in changing one of the SCTP port numbers during the processing, which requires the recomputation of the transport layer checksum by the NAPT function. Whereas for UDP and TCP this can be done very efficiently, for SCTP the checksum (CRC32c) over the entire packet needs to be recomputed (see Appendix B of [RFC4960] for details of the CRC32c computation). This would considerably add to the NAT computational burden, however hardware support can mitigate this in some implementations.

An SCTP endpoint can have multiple addresses but only has a single port number to use. To make multipoint traversal work, all the NAT functions involved need to recognize the packets they see as belonging to the same SCTP association and perform port number translation in a consistent way. One possible way of doing this is to use a pre-defined table of port numbers and addresses configured within each NAT function. Other mechanisms could make use of NAT to NAT communication. Such mechanisms have not been deployed on a wide scale base and thus are not a preferred solution. Therefore an SCTP variant of NAT function has been developed (see Section 4.3).

4.3. The SCTP-Specific Variant of NAT

In this section it is allowed that there are multiple SCTP capable hosts behind a NAT function that share one External-Address. Furthermore, this section focuses on the single point traversal scenario (see Section 4.1.1).

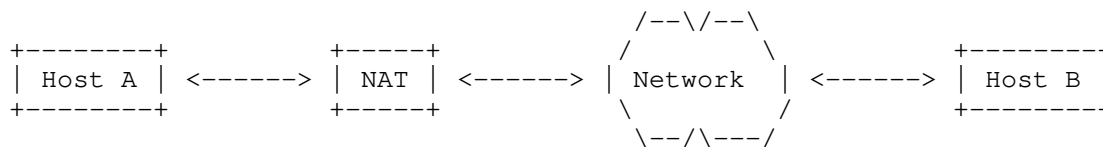
The modification of outgoing SCTP packets sent from an internal host is simple: the source address of the packets has to be replaced with the External-Address. It might also be necessary to establish some state in the NAT function to later handle incoming packets.

Typically, the NAT function has to maintain a NAT binding table of Internal-VTag, Internal-Port, Remote-VTag, Remote-Port, Internal-Address, and whether the restart procedure is disabled or not. An entry in that NAT binding table is called a NAT-State control block. The function Create() obtains the just mentioned parameters and returns a NAT-State control block. A NAT function MAY allow creating NAT-State control blocks via a management interface.

For SCTP packets coming from the external realm of the NAT function the destination address of the packets has to be replaced with the Internal-Address of the host to which the packet has to be delivered, if a NAT state entry is found. The lookup of the Internal-Address is based on the Remote-VTag, Remote-Port, Internal-VTag and the Internal-Port.

The entries in the NAT binding table need to fulfill some uniqueness conditions. There can not be more than one entry NAT binding table with the same pair of Internal-Port and Remote-Port. This rule can be relaxed, if all NAT binding table entries with the same Internal-Port and Remote-Port have the support for the restart procedure disabled (see Section 5.3.1). In this case there can not be no more than one entry with the same Internal-Port, Remote-Port and Remote-VTag and no more than one NAT binding table entry with the same Internal-Port, Remote-Port, and Int-VTag.

The processing of outgoing SCTP packets containing an INIT chunk is illustrated in the following figure. This scenario is valid for all message flows in this section.



```

INIT[Initiate-Tag]
Int-Addr:Int-Port -----> Rem-Addr:Rem-Port
Rem-VTag=0

Create(Initiate-Tag, Int-Port, 0, Rem-Port, Int-Addr,
      IsRestartDisabled)
Returns(NAT-State control block)

```

Translate To:

```

INIT[Initiate-Tag]
Ext-Addr:Int-Port -----> Rem-Addr:Rem-Port
Rem-VTag=0

```

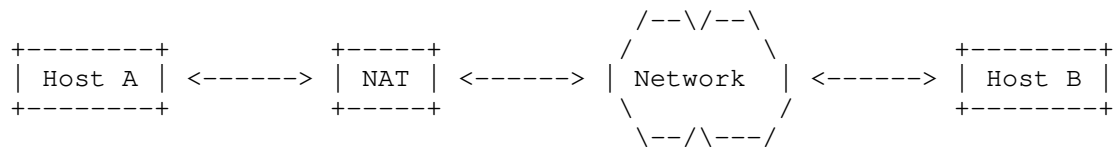
Normally a NAT binding table entry will be created.

However, it is possible that there is already a NAT binding table entry with the same Remote-Port, Internal-Port, and Internal-VTag but different Internal-Address and the restart procedure is disabled. In this case the packet containing the INIT chunk MUST be dropped by the NAT and a packet containing an ABORT chunk SHOULD be sent to the SCTP host that originated the packet with the M bit set and 'VTag and Port Number Collision' error cause (see Section 5.1.1 for the format). The source address of the packet containing the ABORT chunk MUST be the destination address of the packet containing the INIT chunk.

If an outgoing SCTP packet contains an INIT or ASCONF chunk and a matching NAT binding table entry is found, the packet is processed as a normal outgoing packet.

It is also possible that a NAT binding table entry with the same Remote-Port and Internal-Port exists without an Internal-VTag conflict but there exists a NAT binding table entry with the same port numbers but a different Internal-Address and the restart procedure is not disabled. In such a case the packet containing the INIT chunk MUST be dropped by the NAT function and a packet containing an ABORT chunk SHOULD be sent to the SCTP host that originated the packet with the M bit set and 'Port Number Collision' error cause (see Section 5.1.1 for the format).

The processing of outgoing SCTP packets containing no INIT chunks is described in the following figure.

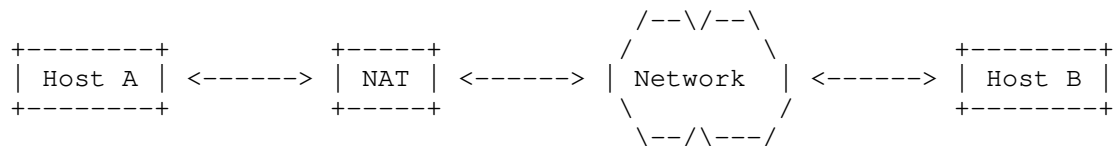


Int-Addr:Int-Port -----> Rem-Addr:Rem-Port
 Rem-VTag

Translate To:

Ext-Addr:Int-Port -----> Rem-Addr:Rem-Port
 Rem-VTag

The processing of incoming SCTP packets containing an INIT ACK chunk is illustrated in the following figure. The Lookup() function has as input the Internal-VTag, Internal-Port, Remote-VTag, and Remote-Port. It returns the corresponding entry of the NAT binding table and updates the Remote-VTag by substituting it with the value of the Initiate-Tag of the INIT ACK chunk. The wildcard character signifies that the parameter's value is not considered in the Lookup() function or changed in the Update() function, respectively.



INIT ACK[Initiate-Tag]
 Ext-Addr:Int-Port <---- Rem-Addr:Rem-Port
 Int-VTag

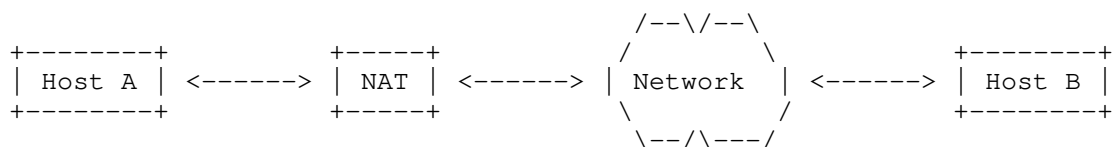
Lookup(Int-VTag, Int-Port, *, Rem-Port)
 Update(*, *, Initiate-Tag, *)

Returns(NAT-State control block containing Int-Addr)

INIT ACK[Initiate-Tag]
 Int-Addr:Int-Port <----- Rem-Addr:Rem-Port
 Int-VTag

In the case where the Lookup function fails because it does not find an entry, the SCTP packet is dropped. If it succeeds, the Update routine inserts the Remote-VTag (the Initiate-Tag of the INIT ACK chunk) in the NAT-State control block.

The processing of incoming SCTP packets containing an ABORT or SHUTDOWN COMPLETE chunk with the T bit set is illustrated in the following figure.



Ext-Addr:Int-Port <----- Rem-Addr:Rem-Port
Rem-VTag

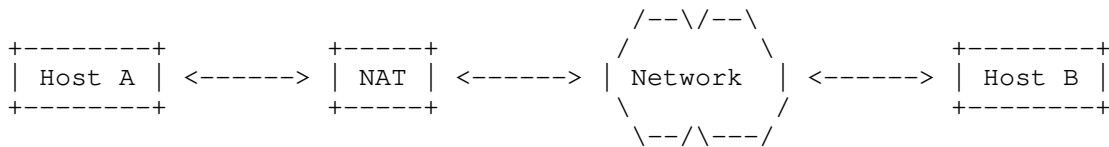
Lookup(*, Int-Port, Rem-VTag, Rem-Port)

Returns (NAT-State control block containing Int-Addr)

Int-Addr:Int-Port <----- Rem-Addr:Rem-Port
Rem-VTag

For an incoming packet containing an INIT chunk a table lookup is made only based on the addresses and port numbers. If an entry with a Remote-VTag of zero is found, it is considered a match and the Remote-VTag is updated. If an entry with a non-matching Remote-VTag is found or no entry is found, the incoming packet is silently dropped. If an entry with a matching Remote-VTag is found, the incoming packet is forwarded. This allows the handling of INIT collision through NAT functions.

The processing of other incoming SCTP packets is described in the following figure.



Ext-Addr: Int-Port <----- Rem-Addr: Rem-Port
Int-VTag

Lookup(Int-VTag, Int-Port, *, Rem-Port)

Returns(NAT-State control block containing Internal-Address)

Int-Addr: Int-Port <----- Rem-Addr: Rem-Port
Int-VTag

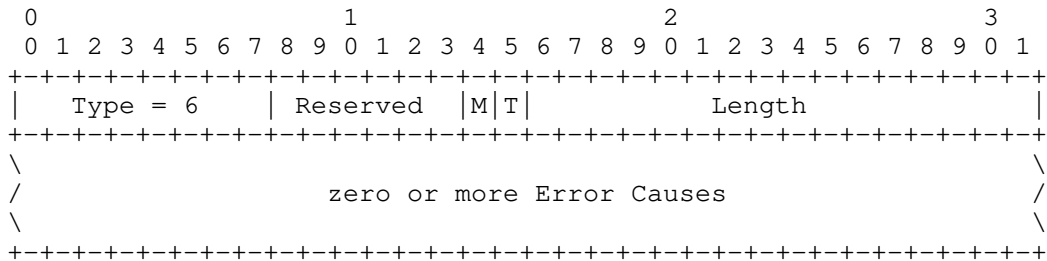
5. Data Formats

This section defines the formats used to support NAT traversal. Section 5.1 and Section 5.2 describe chunks and error causes sent by NAT functions and received by SCTP endpoints. Section 5.3 describes parameters sent by SCTP endpoints and used by NAT functions and SCTP endpoints.

5.1. Modified Chunks

This section presents existing chunks defined in [RFC4960] for which additional flags are specified by this document.

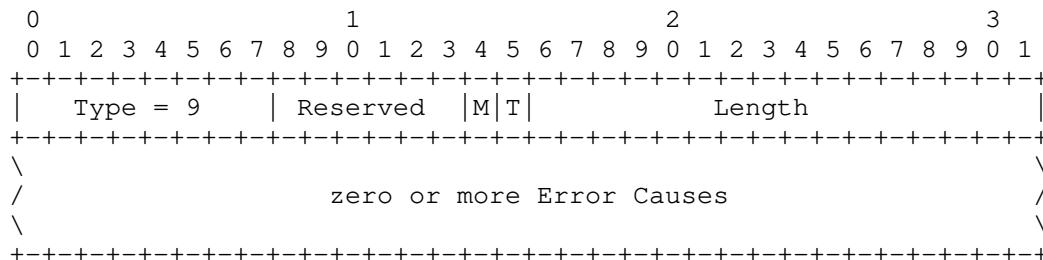
5.1.1. Extended ABORT Chunk



The ABORT chunk is extended to add the new 'M bit'. The M bit indicates to the receiver of the ABORT chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box (e.g., NAT).

[NOTE to RFC-Editor: Assignment of M bit to be confirmed by IANA.]

5.1.2. Extended ERROR Chunk



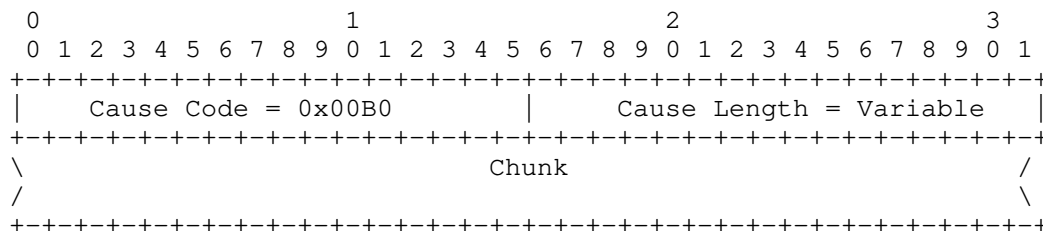
The ERROR chunk defined in [RFC4960] is extended to add the new 'M bit'. The M bit indicates to the receiver of the ERROR chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box.

[NOTE to RFC-Editor: Assignment of M bit to be confirmed by IANA.]

5.2. New Error Causes

This section defines the new error causes added by this document.

5.2.1. VTag and Port Number Collision Error Cause



Cause Code: 2 bytes (unsigned integer)

This field holds the IANA defined cause code for the 'VTag and Port Number Collision' Error Cause. IANA is requested to assign the value 0x00B0 for this cause code.

Cause Length: 2 bytes (unsigned integer)

This field holds the length in bytes of the error cause. The value **MUST** be the length of the Cause-Specific Information plus 4.

Chunk: variable length

The Cause-Specific Information is filled with the chunk that caused this error. This can be an INIT, INIT ACK, or ASCONF chunk. Note that if the entire chunk will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

[NOTE to RFC-Editor: Assignment of cause code to be confirmed by IANA.]

5.2.2. Missing State Error Cause

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Cause Code = 0x00B1										Cause Length = Variable																													
Original Packet																																							

Cause Code: 2 bytes (unsigned integer)

This field holds the IANA defined cause code for the 'Missing State' Error Cause. IANA is requested to assign the value 0x00B1 for this cause code.

Cause Length: 2 bytes (unsigned integer)

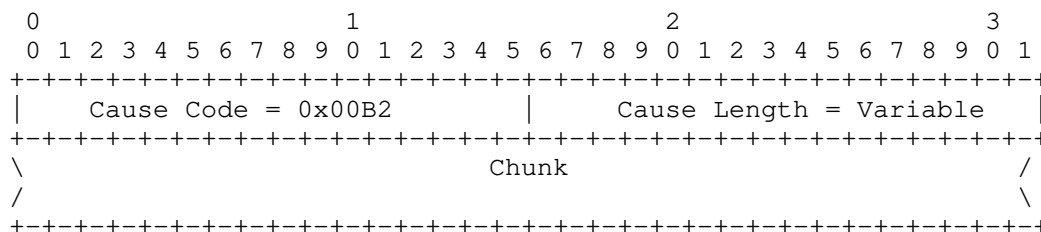
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

Original Packet: variable length

The Cause-Specific Information is filled with the IPv4 or IPv6 packet that caused this error. The IPv4 or IPv6 header MUST be included. Note that if the packet will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

[NOTE to RFC-Editor: Assignment of cause code to be confirmed by IANA.]

5.2.3. Port Number Collision Error Cause



Cause Code: 2 bytes (unsigned integer)

This field holds the IANA defined cause code for the 'Port Number Collision' Error Cause. IANA is requested to assign the value 0x00B2 for this cause code.

Cause Length: 2 bytes (unsigned integer)

This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

Chunk: variable length

The Cause-Specific Information is filled with the chunk that caused this error. This can be an INIT, INIT ACK, or ASCONF chunk. Note that if the entire chunk will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

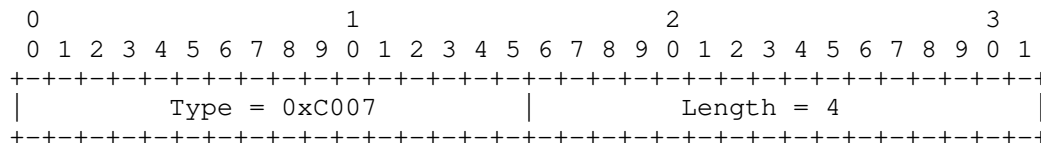
[NOTE to RFC-Editor: Assignment of cause code to be confirmed by IANA.]

5.3. New Parameters

This section defines new parameters and their valid appearance defined by this document.

5.3.1. Disable Restart Parameter

This parameter is used to indicate that the restart procedure is requested to be disabled. Both endpoints of an association MUST include this parameter in the INIT chunk and INIT ACK chunk when establishing an association and MUST include it in the ASCONF chunk when adding an address to successfully disable the restart procedure.



Parameter Type: 2 bytes (unsigned integer)

This field holds the IANA defined parameter type for the Disable Restart Parameter. IANA is requested to assign the value 0xC007 for this parameter type.

Parameter Length: 2 bytes (unsigned integer)

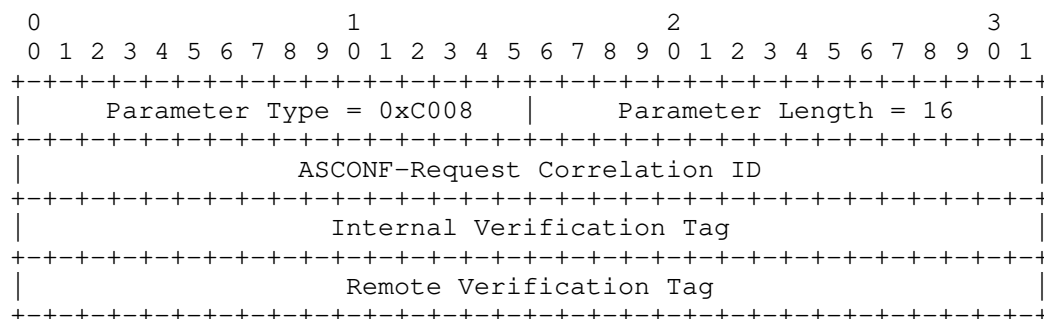
This field holds the length in bytes of the parameter. The value MUST be 4.

[NOTE to RFC-Editor: Assignment of parameter type to be confirmed by IANA.]

The Disable Restart Parameter MAY appear in INIT, INIT ACK and ASCONF chunks and MUST NOT appear in any other chunk.

5.3.2. VTags Parameter

This parameter is used to help a NAT function to recover from state loss.



Parameter Type: 2 bytes (unsigned integer)

This field holds the IANA defined parameter type for the VTags Parameter. IANA is requested to assign the value 0xC008 for this parameter type.

Parameter Length: 2 bytes (unsigned integer)

This field holds the length in bytes of the parameter. The value MUST be 16.

ASCONF-Request Correlation ID: 4 bytes (unsigned integer)

This is an opaque integer assigned by the sender to identify each request parameter. The receiver of the ASCONF Chunk will copy this 32-bit value into the ASCONF Response Correlation ID field of the ASCONF ACK response parameter. The sender of the packet containing the ASCONF chunk can use this same value in the ASCONF ACK chunk to find which request the response is for. The receiver MUST NOT change the value of the ASCONF-Request Correlation ID.

Internal Verification Tag: 4 bytes (unsigned integer)

The Verification Tag that the internal host has chosen for the association. The Verification Tag is a unique 32-bit tag that accompanies any incoming SCTP packet for this association to the Internal-Address.

Remote Verification Tag: 4 bytes (unsigned integer)

The Verification Tag that the host holding the Remote-Address has chosen for the association. The VTag is a unique 32-bit tag that accompanies any outgoing SCTP packet for this association to the Remote-Address.

[NOTE to RFC-Editor: Assignment of parameter type to be confirmed by IANA.]

The VTags Parameter MAY appear in ASCONF chunks and MUST NOT appear in any other chunk.

6. Procedures for SCTP Endpoints and NAT Functions

If an SCTP endpoint is behind an SCTP-aware NAT, a number of problems can arise as it tries to communicate with its peers:

- * IP addresses can not be included in the SCTP packet. This is discussed in Section 6.1.
- * More than one host behind a NAT function could select the same VTag and source port number when communicating with the same peer server. This creates a situation where the NAT function will not be able to tell the two associations apart. This situation is discussed in Section 6.2.
- * If an SCTP endpoint is a server communicating with multiple peers and the peers are behind the same NAT function, then these peers cannot be distinguished by the server. This case is discussed in Section 6.3.
- * A restart of a NAT function during a conversation could cause a loss of its state. This problem and its solution is discussed in Section 6.4.
- * NAT functions need to deal with SCTP packets being fragmented at the IP layer. This is discussed in Section 6.5.
- * An SCTP endpoint can be behind two NAT functions in parallel providing redundancy. The method to set up this scenario is discussed in Section 6.6.

The mechanisms to solve these problems require additional chunks and parameters, defined in this document, and modified handling procedures from those specified in [RFC4960] as described below.

6.1. Association Setup Considerations for Endpoints

The association setup procedure defined in [RFC4960] allows multi-homed SCTP endpoints to exchange its IP-addresses by using IPv4 or IPv6 address parameters in the INIT and INIT ACK chunks. However, this does not work when NAT functions are present.

Every association setup from a host behind a NAT function MUST NOT use multiple internal addresses. The INIT chunk MUST NOT contain an IPv4 Address parameter, IPv6 Address parameter, or Supported Address Types parameter. The INIT ACK chunk MUST NOT contain any IPv4 Address parameter or IPv6 Address parameter using non-global addresses. The INIT chunk and the INIT ACK chunk MUST NOT contain any Host Name parameters.

If the association is intended to be finally multi-homed, the procedure in Section 6.6 MUST be used.

The INIT and INIT ACK chunk SHOULD contain the Disable Restart parameter defined in Section 5.3.1.

6.2. Handling of Internal Port Number and Verification Tag Collisions

Consider the case where two hosts in the Internal-Address space want to set up an SCTP association with the same service provided by some remote hosts. This means that the Remote-Port is the same. If they both choose the same Internal-Port and Internal-VTag, the NAT function cannot distinguish between incoming packets anymore. However, this is unlikely. The Internal-VTags are chosen at random and if the Internal-Ports are also chosen from the ephemeral port range at random (see [RFC6056]) this gives a 46-bit random number that has to match.

The same can happen with the Remote-VTag when a packet containing an INIT ACK chunk or an ASCONF chunk is processed by the NAT function.

6.2.1. NAT Function Considerations

If the NAT function detects a collision of internal port numbers and verification tags, it SHOULD send a packet containing an ABORT chunk with the M bit set if the collision is triggered by a packet containing an INIT or INIT ACK chunk. If such a collision is triggered by a packet containing an ASCONF chunk, it SHOULD send a packet containing an ERROR chunk with the M bit. The M bit is a new

bit defined by this document to express to SCTP that the source of this packet is a "middle" box, not the peer SCTP endpoint (see Section 5.1.1). If a packet containing an INIT ACK chunk triggers the collision, the corresponding packet containing the ABORT chunk MUST contain the same source and destination address and port numbers as the packet containing the INIT ACK chunk. If a packet containing an INIT chunk or an ASCONF chunk, the source and destination address and port numbers MUST be swapped.

The sender of the packet containing an ERROR or ABORT chunk MUST include the error cause with cause code 'VTag and Port Number Collision' (see Section 5.2.1).

6.2.2. Endpoint Considerations

The sender of the packet containing the INIT chunk or the receiver of a packet containing the INIT ACK chunk, upon reception of a packet containing an ABORT chunk with M bit set and the appropriate error cause code for colliding NAT binding table state is included, SHOULD reinitiate the association setup procedure after choosing a new initiate tag, if the association is in COOKIE-WAIT state. In any other state, the SCTP endpoint MUST NOT respond.

The sender of the packet containing the ASCONF chunk, upon reception of a packet containing an ERROR chunk with M bit set, MUST stop adding the path to the association.

6.3. Handling of Internal Port Number Collisions

When two SCTP hosts are behind an SCTP-aware NAT it is possible that two SCTP hosts in the Internal-Address space will want to set up an SCTP association with the same server running on the same remote host. If the two hosts choose the same internal port, this is considered an internal port number collision.

For the NAT function, appropriate tracking can be performed by assuring that the VTags are unique between the two hosts.

6.3.1. NAT Function Considerations

The NAT function, when processing the packet containing the INIT ACK chunk, SHOULD note in its NAT binding table if the association supports the disable restart extension. This note is used when establishing future associations (i.e. when processing a packet containing an INIT chunk from an internal host) to decide if the connection can be allowed. The NAT function does the following when processing a packet containing an INIT chunk:

- * If the packet containing the INIT chunk is originating from an internal port to a remote port for which the NAT function has no matching NAT binding table entry, it MUST allow the packet containing the INIT chunk creating an NAT binding table entry.
- * If the packet containing the INIT chunk matches an existing NAT binding table entry, it MUST validate that the disable restart feature is supported and, if it does, allow the packet containing the INIT chunk to be forwarded.
- * If the disable restart feature is not supported, the NAT function SHOULD send a packet containing an ABORT chunk with the M bit set.

The 'Port Number Collision' error cause (see Section 5.2.3) MUST be included in the ABORT chunk sent in response to the packet containing an INIT chunk.

If the collision is triggered by a packet containing an ASCONF chunk, a packet containing an ERROR chunk with the 'Port Number Collision' error cause SHOULD be sent in response to the packet containing the ASCONF chunk.

6.3.2. Endpoint Considerations

For the remote SCTP server this means that the Remote-Port and the Remote-Address are the same. If they both have chosen the same Internal-Port the server cannot distinguish between both associations based on the address and port numbers. For the server it looks like the association is being restarted. To overcome this limitation the client sends a Disable Restart parameter in the INIT chunk.

When the server receives this parameter it does the following:

- * It MUST include a Disable Restart parameter in the INIT ACK to inform the client that it will support the feature.
- * It MUST disable the restart procedures defined in [RFC4960] for this association.

Servers that support this feature will need to be capable of maintaining multiple connections to what appears to be the same peer (behind the NAT function) differentiated only by the VTags.

6.4. Handling of Missing State

6.4.1. NAT Function Considerations

If the NAT function receives a packet from the internal network for which the lookup procedure does not find an entry in the NAT binding table, a packet containing an ERROR chunk SHOULD be sent back with the M bit set. The source address of the packet containing the ERROR chunk MUST be the destination address of the packet received from the internal network. The verification tag is reflected and the T bit is set. Such a packet containing an ERROR chunk SHOULD NOT be sent if the received packet contains an ASCONF chunk with the VTags parameter or an ABORT, SHUTDOWN COMPLETE or INIT ACK chunk. A packet containing an ERROR chunk MUST NOT be sent if the received packet contains an ERROR chunk with the M bit set. In any case, the packet SHOULD NOT be forwarded to the remote address.

If the NAT function receives a packet from the internal network for which it has no NAT binding table entry and the packet contains an ASCONF chunk with the VTags parameter, the NAT function MUST update its NAT binding table according to the verification tags in the VTags parameter and, if present, the Disable Restart parameter.

When sending a packet containing an ERROR chunk, the error cause 'Missing State' (see Section 5.2.2) MUST be included and the M bit of the ERROR chunk MUST be set (see Section 5.1.2).

6.4.2. Endpoint Considerations

Upon reception of this packet containing the ERROR chunk by an SCTP endpoint the receiver takes the following actions:

- * It SHOULD validate that the verification tag is reflected by looking at the VTag that would have been included in an outgoing packet. If the validation fails, discard the received packet containing the ERROR chunk.
- * It SHOULD validate that the peer of the SCTP association supports the dynamic address extension. If the validation fails, discard the received packet containing the ERROR chunk.
- * It SHOULD generate a packet containing a new ASCONF chunk containing the VTags parameter (see Section 5.3.2) and the Disable Restart parameter (see Section 5.3.1) if the association is using the disable restart feature. By processing this packet the NAT function can recover the appropriate state. The procedures for generating an ASCONF chunk can be found in [RFC5061].

The peer SCTP endpoint receiving such a packet containing an ASCONF chunk SHOULD add the address and respond with an acknowledgment if the address is new to the association (following all procedures defined in [RFC5061]). If the address is already part of the association, the SCTP endpoint MUST NOT respond with an error, but instead SHOULD respond with a packet containing an ASCONF ACK chunk acknowledging the address and take no action (since the address is already in the association).

Note that it is possible that upon receiving a packet containing an ASCONF chunk containing the VTags parameter the NAT function will realize that it has an 'Internal Port Number and Verification Tag collision'. In such a case the NAT function SHOULD send a packet containing an ERROR chunk with the error cause code set to 'VTag and Port Number Collision' (see Section 5.2.1).

If an SCTP endpoint receives a packet containing an ERROR chunk with 'Internal Port Number and Verification Tag collision' as the error cause and the packet in the Error Chunk contains an ASCONF with the VTags parameter, careful examination of the association is necessary. The endpoint does the following:

- * It MUST validate that the verification tag is reflected by looking at the VTag that would have been included in the outgoing packet. If the validation fails, it MUST discard the packet.
- * It MUST validate that the peer of the SCTP association supports the dynamic address extension. If the peer does not support this extension, it MUST discard the received packet containing the ERROR chunk.
- * If the association is attempting to add an address (i.e. following the procedures in Section 6.6) then the endpoint MUST NOT consider the address part of the association and SHOULD make no further attempt to add the address (i.e. cancel any ASCONF timers and remove any record of the path), since the NAT function has a VTag collision and the association cannot easily create a new VTag (as it would if the error occurred when sending a packet containing an INIT chunk).
- * If the endpoint has no other path, i.e. the procedure was executed due to missing a state in the NAT function, then the endpoint MUST abort the association. This would occur only if the local NAT function restarted and accepted a new association before attempting to repair the missing state (Note that this is no different than what happens to all TCP connections when a NAT function loses its state).

6.5. Handling of Fragmented SCTP Packets by NAT Functions

SCTP minimizes the use of IP-level fragmentation. However, it can happen that using IP-level fragmentation is needed to continue an SCTP association. For example, if the path MTU is reduced and there are still some DATA chunk in flight, which require packets larger than the new path MTU. If IP-level fragmentation can not be used, the SCTP association will be terminated in a non-graceful way. See [RFC8900] for more information about IP fragmentation.

Therefore, a NAT function MUST be able to handle IP-level fragmented SCTP packets. The fragments MAY arrive in any order.

When an SCTP packet can not be forwarded by the NAT function due to MTU issues and the IP header forbids fragmentation, the NAT MUST send back a "Fragmentation needed and DF set" ICMPv4 or PTB ICMPv6 message to the internal host. This allows for a faster recovery from this packet drop.

6.6. Multi Point Traversal Considerations for Endpoints

If a multi-homed SCTP endpoint behind a NAT function connects to a peer, it MUST first set up the association single-homed with only one address causing the first NAT function to populate its state. Then it SHOULD add each IP address using packets containing ASCONF chunks sent via their respective NAT functions. The address used in the Add IP address parameter is the wildcard address (0.0.0.0 or ::0) and the address parameter in the ASCONF chunk SHOULD also contain the VTags parameter and optionally the Disable Restart parameter.

7. SCTP NAT YANG Module

This section defines a YANG module for SCTP NAT.

The terminology for describing YANG data models is defined in [RFC7950]. The meaning of the symbols in tree diagrams is defined in [RFC8340].

7.1. Tree Structure

This module augments NAT YANG module [RFC8512] with SCTP specifics. The module supports both classical SCTP NAT (that is, rewrite port numbers) and SCTP-specific variant where the ports numbers are not altered. The YANG "feature" is used to indicate whether SCTP-specific variant is supported.

The tree structure of the SCTP NAT YANG module is provided below:

```

module: ietf-nat-sctp
  augment /nat:nat/nat:instances/nat:instance
    /nat:policy/nat:timers:
      +--rw sctp-timeout?  uint32
  augment /nat:nat/nat:instances/nat:instance
    /nat:mapping-table/nat:mapping-entry:
      +--rw int-VTag?      uint32 {sctp-nat}?
      +--rw rem-VTag?      uint32 {sctp-nat}?

```

Concretely, the SCTP NAT YANG module augments the NAT YANG module (policy, in particular) with the following:

- * The sctp-timeout is used to control the SCTP inactivity timeout. That is, the time an SCTP mapping will stay active without SCTP packets traversing the NAT. This timeout can be set only for SCTP. Hence, `"/nat:nat/nat:instances/nat:instance/nat:policy/nat:transport-protocols/nat:protocol-id"` MUST be set to `'132'` (SCTP).

In addition, the SCTP NAT YANG module augments the mapping entry with the following parameters defined in Section 3. These parameters apply only for SCTP NAT mapping entries (i.e., `"/nat/instances/instance/mapping-table/mapping-entry/transport-protocol"` MUST be set to `'132'`);

- * The Internal Verification Tag (Int-VTag)
- * The Remote Verification Tag (Rem-VTag)

7.2. YANG Module

```

<CODE BEGINS> file "ietf-nat-sctp@2020-11-02.yang"
module ietf-nat-sctp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nat-sctp";
  prefix nat-sctp;

  import ietf-nat {
    prefix nat;
    reference
      "RFC 8512: A YANG Module for Network Address Translation
       (NAT) and Network Prefix Translation (NPT)";
  }

  organization
    "IETF TSVWG Working Group";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/tsvwg/>

```

WG List: <mailto:tsvwg@ietf.org>

Author: Mohamed Boucadair
<mailto:mohamed.boucadair@orange.com>;

description

"This module augments NAT YANG module with Stream Control Transmission Protocol (SCTP) specifics. The extension supports both a classical SCTP NAT (that is, rewrite port numbers) and a, SCTP-specific variant where the ports numbers are not altered.

Copyright (c) 2020 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

revision 2019-11-18 {

description

"Initial revision.";

reference

"RFC XXXX: Stream Control Transmission Protocol (SCTP) Network Address Translation Support";

}

feature sctp-nat {

description

"This feature means that SCTP-specific variant of NAT is supported. That is, avoid rewriting port numbers.";

reference

"Section 4.3 of RFC XXXX.";

}

augment "/nat:nat/nat:instances/nat:instance"

+ "/nat:policy/nat:timers" {

when "/nat:nat/nat:instances/nat:instance"

+ "/nat:policy/nat:transport-protocols"

+ "/nat:protocol-id = 132";

description

"Extends NAT policy with a timeout for SCTP mapping entries.";

```
    leaf sctp-timeout {
      type uint32;
      units "seconds";
      description
        "SCTP inactivity timeout. That is, the time an SCTP
        mapping entry will stay active without packets
        traversing the NAT.";
    }
  }

  augment "/nat:nat/nat:instances/nat:instance"
    + "/nat:mapping-table/nat:mapping-entry" {
    when "nat:transport-protocol = 132";
    if-feature "sctp-nat";
    description
      "Extends the mapping entry with SCTP specifics.";

    leaf int-VTag {
      type uint32;
      description
        "The Internal Verification Tag that the internal
        host has chosen for this communication.";
    }
    leaf rem-VTag {
      type uint32;
      description
        "The Remote Verification Tag that the remote
        peer has chosen for this communication.";
    }
  }
}
<CODE ENDS>
```

8. Various Examples of NAT Traversals

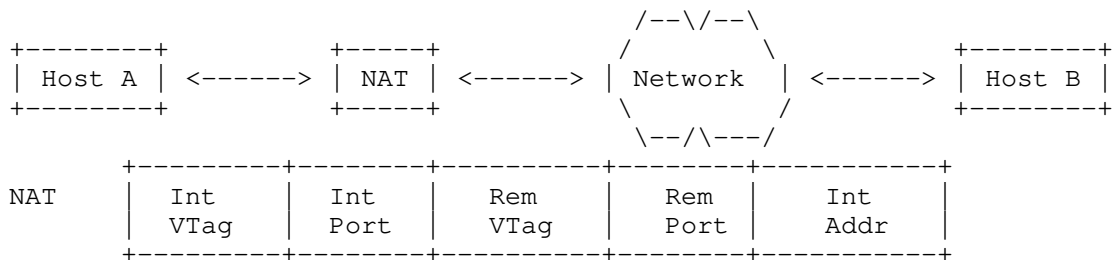
Please note that this section is informational only.

The addresses being used in the following examples are IPv4 addresses for private-use networks and for documentation as specified in [RFC6890]. However, the method described here is not limited to this NAT44 case.

The NAT binding table entries shown in the following examples do not include the flag indicating whether the restart procedure is supported or not. This flag is not relevant for these examples.

8.1. Single-homed Client to Single-homed Server

The internal client starts the association with the remote server via a four-way-handshake. Host A starts by sending a packet containing an INIT chunk.



```

INIT[Initiate-Tag = 1234]
10.0.0.1:1 -----> 203.0.113.1:2
    Rem-VTtag = 0

```

A NAT binding tabled entry is created, the source address is substituted and the packet is sent on:

NAT function creates entry:

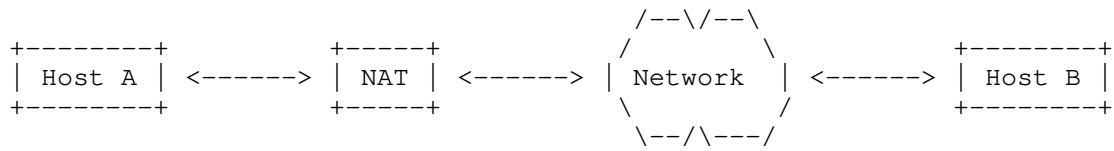
NAT	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	0	2	10.0.0.1

```

                                INIT[Initiate-Tag = 1234]
192.0.2.1:1 -----> 203.0.113.1:2
                                Rem-VTtag = 0

```

Host B receives the packet containing an INIT chunk and sends a packet containing an INIT ACK chunk with the NAT's Remote-address as destination address.



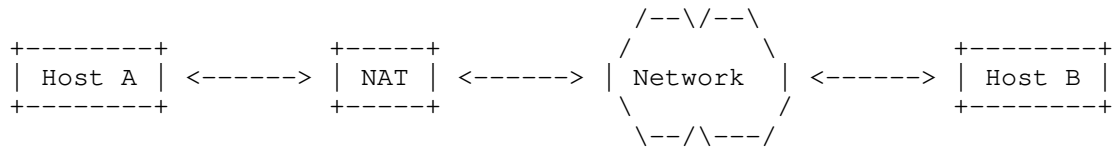
INIT ACK[Initiate-Tag = 5678]
 192.0.2.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

NAT function updates entry:

NAT	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	5678	2	10.0.0.1

INIT ACK[Initiate-Tag = 5678]
 10.0.0.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

The handshake finishes with a COOKIE ECHO acknowledged by a COOKIE ACK.



COOKIE ECHO
 10.0.0.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

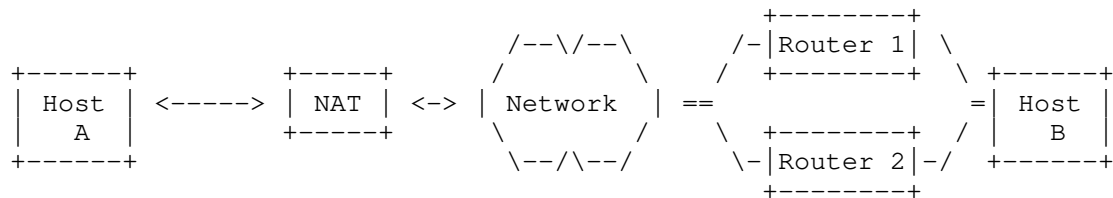
COOKIE ECHO
 192.0.2.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

COOKIE ACK
 192.0.2.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

COOKIE ACK
 10.0.0.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

8.2. Single-homed Client to Multi-homed Server

The internal client is single-homed whereas the remote server is multi-homed. The client (Host A) sends a packet containing an INIT chunk like in the single-homed case.



NAT	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
-----	-------------	-------------	-------------	-------------	-------------

```

INIT[Initiate-Tag = 1234]
10.0.0.1:1 ---> 203.0.113.1:2
Rem-VTag = 0
  
```

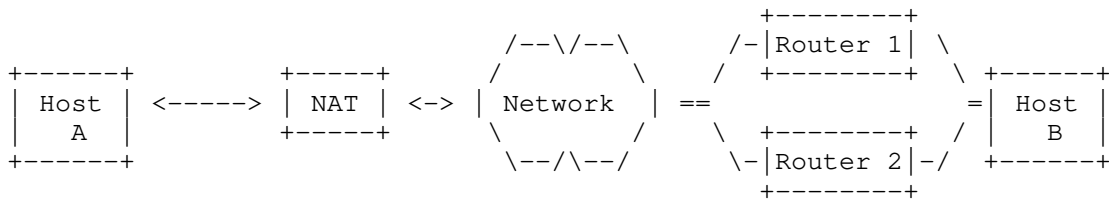
NAT function creates entry:

NAT	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	0	2	10.0.0.1

```

INIT[Initiate-Tag = 1234]
192.0.2.1:1 -----> 203.0.113.1:2
Rem-VTag = 0
  
```

The server (Host B) includes its two addresses in the INIT ACK chunk.



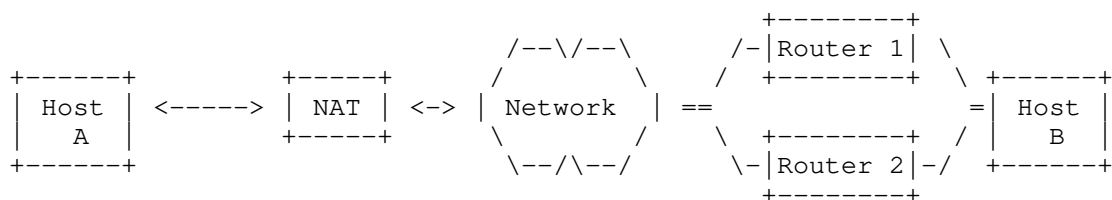
```
INIT ACK[Initiate-tag = 5678, IP-Addr = 203.0.113.129]
192.0.2.1:1 <----- 203.0.113.1:2
                        Int-VTag = 1234
```

The NAT function does not need to change the NAT binding table for the second address:

NAT					
	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	5678	2	10.0.0.1

```
INIT ACK[Initiate-Tag = 5678]
10.0.0.1:1 <--- 203.0.113.1:2
      Int-VTag = 1234
```

The handshake finishes with a COOKIE ECHO acknowledged by a COOKIE ACK.



COOKIE ECHO
 10.0.0.1:1 ---> 203.0.113.1:2
 Rem-VTag = 5678

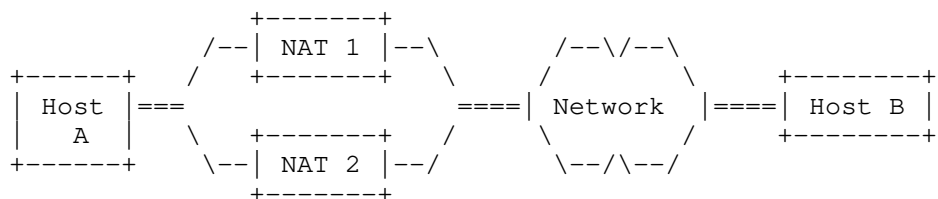
COOKIE ECHO
 192.0.2.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

COOKIE ACK
 192.0.2.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

COOKIE ACK
 10.0.0.1:1 <--- 203.0.113.1:2
 Int-VTag = 1234

8.3. Multihomed Client and Server

The client (Host A) sends a packet containing an INIT chunk to the server (Host B), but does not include the second address.



NAT 1	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr

INIT[Initiate-Tag = 1234]
 10.0.0.1:1 -----> 203.0.113.1:2
 Rem-VTag = 0

NAT function 1 creates entry:

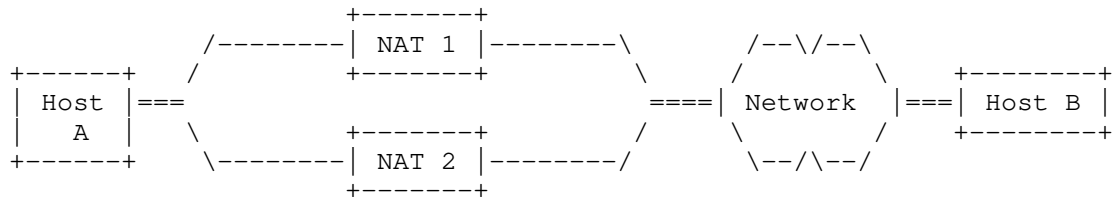
NAT 1	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	0	2	10.0.0.1

```

                                INIT[Initiate-Tag = 1234]
192.0.2.1:1 -----> 203.0.113.1:2
                                Rem-VTag = 0

```

Host B includes its second address in the INIT ACK.



```

INIT ACK[Initiate-Tag = 5678, IP-Addr = 203.0.113.129]
192.0.2.1:1 <----- 203.0.113.1:2
                                Int-VTag = 1234

```

NAT function 1 does not need to update the NAT binding table for the second address:

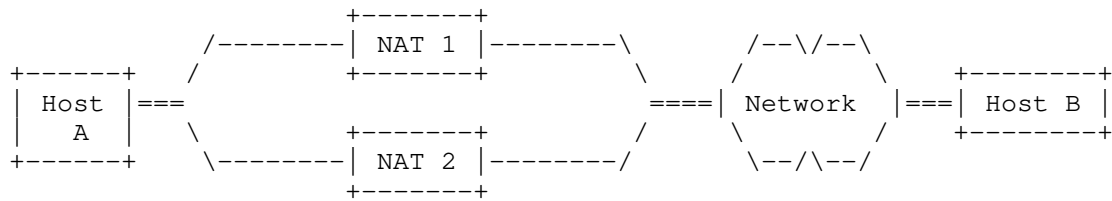
NAT 1	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	5678	2	10.0.0.1

```

INIT ACK[Initiate-Tag = 5678]
10.0.0.1:1 <----- 203.0.113.1:2
                                Int-VTag = 1234

```

The handshake finishes with a COOKIE ECHO acknowledged by a COOKIE ACK.



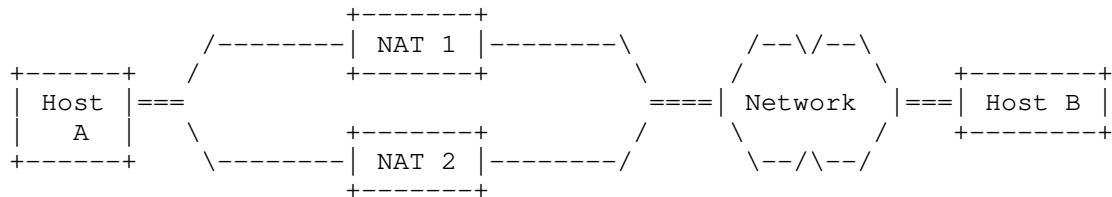
COOKIE ECHO
 10.0.0.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

COOKIE ECHO
 192.0.2.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

COOKIE ACK
 192.0.2.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

COOKIE ACK
 10.0.0.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

Host A announces its second address in an ASCONF chunk. The address parameter contains a wildcard address (0.0.0.0 or ::0) to indicate that the source address has to be added. The address parameter within the ASCONF chunk will also contain the pair of VTags (remote and internal) so that the NAT function can populate its NAT binding table entry completely with this single packet.



ASCONF [ADD-IP=0.0.0.0, INT-VTag=1234, Rem-VTag = 5678]
 10.1.0.1:1 -----> 203.0.113.129:2
 Rem-VTag = 5678

NAT function 2 creates a complete entry:

NAT 2	+-----+				
	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	1234	1	5678	2	10.1.0.1
	+-----+				

```

ASCONF [ADD-IP, Int-VTag=1234, Rem-VTag = 5678]
192.0.2.129:1 -----> 203.0.113.129:2
                        Rem-VTag = 5678

```

```

                        ASCONF ACK
192.0.2.129:1 <----- 203.0.113.129:2
                        Int-VTag = 1234

```

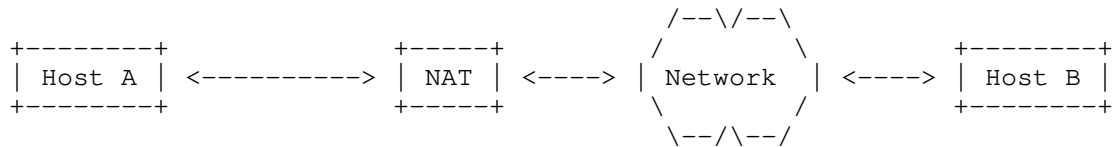
```

                        ASCONF ACK
10.1.0.1:1 <----- 203.0.113.129:2
                        Int-VTag = 1234

```

8.4. NAT Function Loses Its State

Association is already established between Host A and Host B, when the NAT function loses its state and obtains a new external address. Host A sends a DATA chunk to Host B.



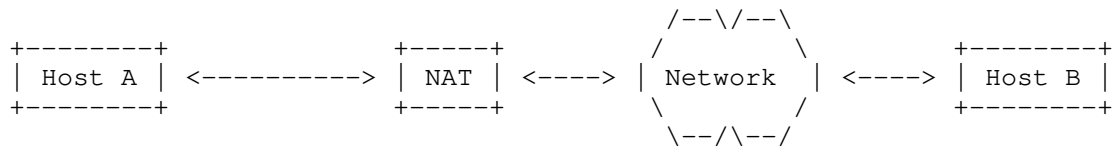
NAT	+-----+				
	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	+-----+				

```

                        DATA
10.0.0.1:1 -----> 203.0.113.1:2
                        Rem-VTag = 5678

```

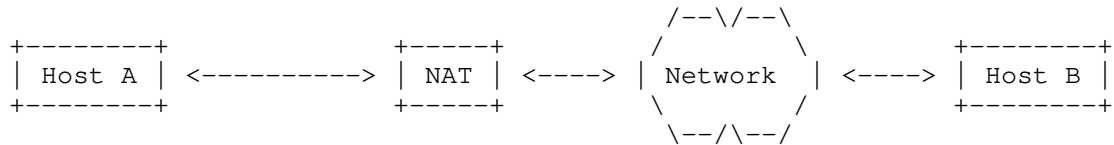
The NAT function cannot find an entry in the NAT binding table for the association. It sends a packet containing an ERROR chunk with the M bit set and the cause "NAT state missing".



```

ERROR [M bit, NAT state missing]
10.0.0.1:1 <----- 203.0.113.1:2
      Rem-VTag = 5678
  
```

On reception of the packet containing the ERROR chunk, Host A sends a packet containing an ASCONF chunk indicating that the former information has to be deleted and the source address of the actual packet added.



```

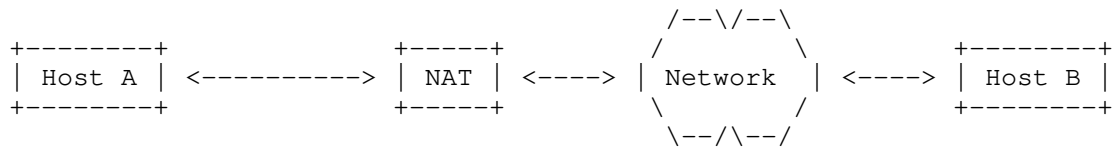
ASCONF [ADD-IP, DELETE-IP, Int-VTag=1234, Rem-VTag = 5678]
10.0.0.1:1 -----> 203.0.113.129:2
      Rem-VTag = 5678
  
```

NAT	+-----+ +-----+ +-----+ +-----+ +-----+				
	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	+-----+ +-----+ +-----+ +-----+ +-----+				
	1234	1	5678	2	10.0.0.1
	+-----+ +-----+ +-----+ +-----+ +-----+				

```

ASCONF [ADD-IP, DELETE-IP, Int-VTag=1234, Rem-VTag = 5678]
      192.0.2.2:1 -----> 203.0.113.129:2
      Rem-VTag = 5678
  
```

Host B adds the new source address to this association and deletes all other addresses from this association.



ASCONF ACK
 192.0.2.2:1 <----- 203.0.113.129:2
 Int-VTag = 1234

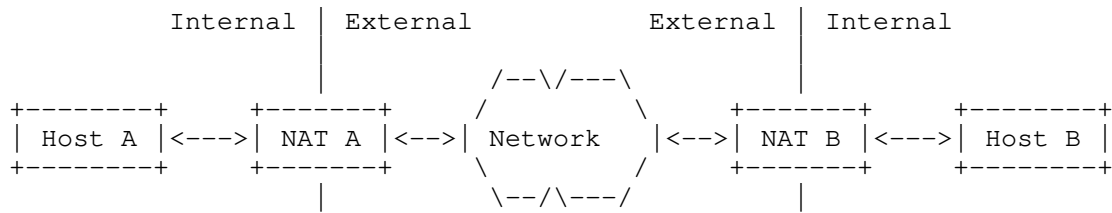
ASCONF ACK
 10.1.0.1:1 <----- 203.0.113.129:2
 Int-VTag = 1234

DATA
 10.0.0.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

DATA
 192.0.2.2:1 -----> 203.0.113.129:2
 Rem-VTag = 5678

8.5. Peer-to-Peer Communications

If two hosts, each of them behind a NAT function, want to communicate with each other, they have to get knowledge of the peer's external address. This can be achieved with a so-called rendezvous server. Afterwards the destination addresses are external, and the association is set up with the help of the INIT collision. The NAT functions create their entries according to their internal peer's point of view. Therefore, NAT function A's Internal-VTag and Internal-Port are NAT function B's Remote-VTag and Remote-Port, respectively. The naming (internal/remote) of the verification tag in the packet flow is done from the sending host's point of view.



NAT Binding Tables

NAT A	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
-------	-------------	-------------	-------------	-------------	-------------

NAT B	Int v-tag	Int port	Rem v-tag	Rem port	Int Addr
-------	--------------	-------------	--------------	-------------	-------------

```

INIT[Initiate-Tag = 1234]
10.0.0.1:1 --> 203.0.113.1:2
    Rem-VTag = 0
  
```

NAT function A creates entry:

NAT A	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	0	2	10.0.0.1

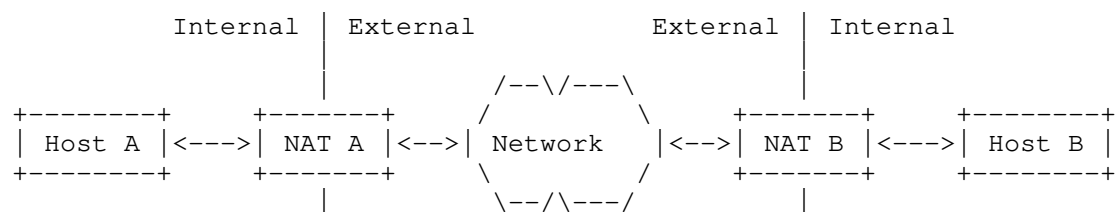
```

INIT[Initiate-Tag = 1234]
192.0.2.1:1 -----> 203.0.113.1:2
    Rem-VTag = 0
  
```

NAT function B processes the packet containing the INIT chunk, but cannot find an entry. The SCTP packet is silently discarded and leaves the NAT binding table of NAT function B unchanged.

NAT B	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
-------	-------------	-------------	-------------	-------------	-------------

Now Host B sends a packet containing an INIT chunk, which is processed by NAT function B. Its parameters are used to create an entry.



```

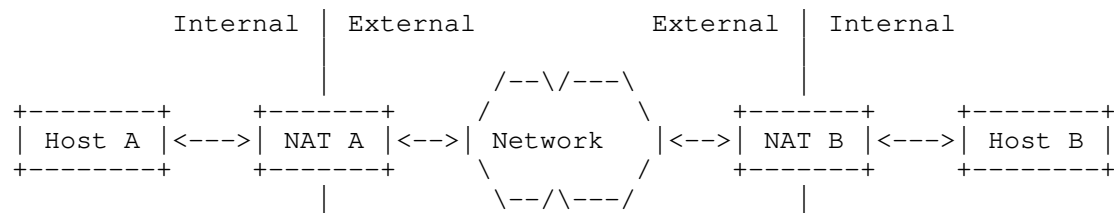
INIT[Initiate-Tag = 5678]
192.0.2.1:1 <-- 10.1.0.1:2
Rem-VTag = 0
  
```

NAT B	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	5678	2	0	1	10.1.0.1

```

INIT[Initiate-Tag = 5678]
192.0.2.1:1 <----- 203.0.113.1:2
Rem-VTag = 0
  
```

NAT function A processes the packet containing the INIT chunk. As the outgoing packet containing an INIT chunk of Host A has already created an entry, the entry is found and updated:

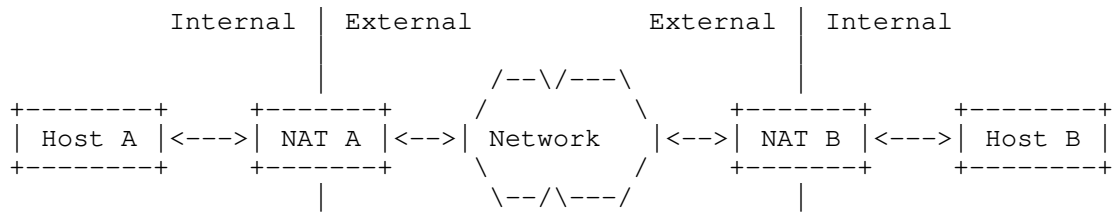


VTag != Int-VTag, but Rem-VTag == 0, find entry.

NAT A	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	5678	2	10.0.0.1

```
INIT[Initiate-tag = 5678]
10.0.0.1:1 <-- 203.0.113.1:2
    Rem-VTag = 0
```

Host A sends a packet containing an INIT ACK chunk, which can pass through NAT function B:



```

INIT ACK[Initiate-Tag = 1234]
10.0.0.1:1 --> 203.0.113.1:2
    Rem-VTag = 5678

```

```

        INIT ACK[Initiate-Tag = 1234]
192.0.2.1:1 -----> 203.0.113.1:2
        Rem-VTag = 5678

```

NAT function B updates entry:

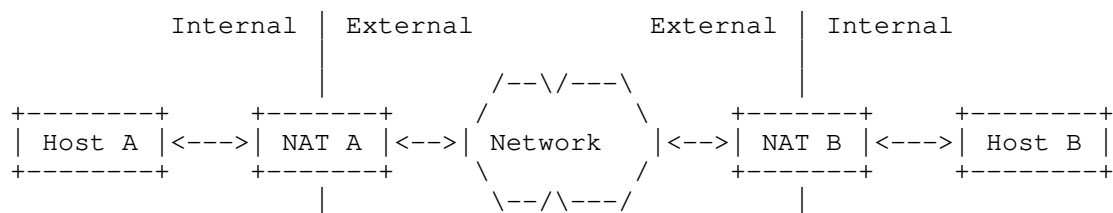
NAT B	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	5678	2	1234	1	10.1.0.1

```

INIT ACK[Initiate-Tag = 1234]
192.0.2.1:1 --> 10.1.0.1:2
    Rem-VTag = 5678

```

The lookup for COOKIE ECHO and COOKIE ACK is successful.



COOKIE ECHO
 192.0.2.1:1 <-- 10.1.0.1:2
 Rem-VTag = 1234

COOKIE ECHO
 192.0.2.1:1 <----- 203.0.113.1:2
 Rem-VTag = 1234

COOKIE ECHO
 10.0.0.1:1 <-- 203.0.113.1:2
 Rem-VTag = 1234

COOKIE ACK
 10.0.0.1:1 --> 203.0.113.1:2
 Rem-VTag = 5678

COOKIE ACK
 192.0.2.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

COOKIE ACK
 192.0.2.1:1 --> 10.1.0.1:2
 Rem-VTag = 5678

9. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to control NAT friendliness.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is extended by supporting one new read/write socket option.

9.1. Get or Set the NAT Friendliness (SCTP_NAT_FRIENDLY)

This socket option uses the option_level IPPROTO_SCTP and the option_name SCTP_NAT_FRIENDLY. It can be used to enable/disable the NAT friendliness for future associations and retrieve the value for future and specific ones.

```
struct sctp_assoc_value {  
    sctp_assoc_t assoc_id;  
    uint32_t assoc_value;  
};
```

assoc_id

This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application can fill in an association identifier or SCTP_FUTURE_ASSOC for this query. It is an error to use SCTP_{CURRENT|ALL}_ASSOC in assoc_id.

assoc_value

A non-zero value indicates a NAT-friendly mode.

10. IANA Considerations

[NOTE to RFC-Editor: "RFCXXXX" is to be replaced by the RFC number you assign this document.]

[NOTE to RFC-Editor: The requested values for the chunk type and the chunk parameter types are tentative and to be confirmed by IANA.]

This document (RFCXXXX) is the reference for all registrations described in this section. The requested changes are described below.

10.1. New Chunk Flags for Two Existing Chunk Types

As defined in [RFC6096] two chunk flags have to be assigned by IANA for the ERROR chunk. The requested value for the T bit is 0x01 and for the M bit is 0x02.

This requires an update of the "ERROR Chunk Flags" registry for SCTP:

ERROR Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	T bit	[RFCXXXX]
0x02	M bit	[RFCXXXX]
0x04	Unassigned	
0x08	Unassigned	
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

Table 2

As defined in [RFC6096] one chunk flag has to be assigned by IANA for the ABORT chunk. The requested value of the M bit is 0x02.

This requires an update of the "ABORT Chunk Flags" registry for SCTP:

ABORT Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	T bit	[RFC4960]
0x02	M bit	[RFCXXXX]
0x04	Unassigned	
0x08	Unassigned	
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

Table 3

10.2. Three New Error Causes

Three error causes have to be assigned by IANA. It is requested to use the values given below.

This requires three additional lines in the "Error Cause Codes" registry for SCTP:

Error Cause Codes

Value	Cause Code	Reference
176	VTag and Port Number Collision	[RFCXXXX]
177	Missing State	[RFCXXXX]
178	Port Number Collision	[RFCXXXX]

Table 4

10.3. Two New Chunk Parameter Types

Two chunk parameter types have to be assigned by IANA. IANA is requested to assign these values from the pool of parameters with the upper two bits set to '11' and to use the values given below.

This requires two additional lines in the "Chunk Parameter Types" registry for SCTP:

Chunk Parameter Types

ID Value	Chunk Parameter Type	Reference
49159	Disable Restart (0xC007)	[RFCXXXX]
49160	VTags (0xC008)	[RFCXXXX]

Table 5

10.4. One New URI

An URI in the "ns" subregistry within the "IETF XML" registry has to be assigned by IANA ([RFC3688]):

URI: urn:ietf:params:xml:ns:yang:ietf-nat-sctp
 Registrant Contact: The IESG.
 XML: N/A; the requested URI is an XML namespace.

10.5. One New YANG Module

An YANG module in the "YANG Module Names" subregistry within the "YANG Parameters" registry has to be assigned by IANA ([RFC6020]):

Name: ietf-nat-sctp
 Namespace: urn:ietf:params:xml:ns:yang:ietf-nat-sctp
 Maintained by IANA: N
 Prefix: nat-sctp
 Reference: RFCXXXX

11. Security Considerations

State maintenance within a NAT function is always a subject of possible Denial Of Service attacks. This document recommends that at a minimum a NAT function runs a timer on any SCTP state so that old association state can be cleaned up.

Generic issues related to address sharing are discussed in [RFC6269] and apply to SCTP as well.

For SCTP endpoints not disabling the restart procedure, this document does not add any additional security considerations to the ones given in [RFC4960], [RFC4895], and [RFC5061].

SCTP endpoints disabling the restart procedure, need to monitor the status of all associations to mitigate resource exhaustion attacks by establishing a lot of associations sharing the same IP addresses and port numbers.

In any case, SCTP is protected by the verification tags and the usage of [RFC4895] against off-path attackers.

For IP-level fragmentation and reassembly related issues see [RFC4963].

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All data nodes defined in the YANG module that can be created, modified, and deleted (i.e., config true, which is the default) are considered sensitive. Write operations (e.g., edit-config) applied to these data nodes without proper protection can negatively affect network operations. An attacker who is able to access the SCTP NAT function can undertake various attacks, such as:

- * Setting a low timeout for SCTP mapping entries to cause failures to deliver incoming SCTP packets.
- * Instantiating mapping entries to cause NAT collision.

12. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, <<https://www.rfc-editor.org/info/rfc4895>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<https://www.rfc-editor.org/info/rfc4960>>.
- [RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <<https://www.rfc-editor.org/info/rfc5061>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, DOI 10.17487/RFC6096, January 2011, <<https://www.rfc-editor.org/info/rfc6096>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <<https://www.rfc-editor.org/info/rfc8512>>.

13. Informative References

- [DOI_10.1145_1496091.1496095]
Hayes, D., But, J., and G. Armitage, "Issues with network address translation for SCTP", ACM SIGCOMM Computer Communication Review Vol. 39, pp. 23-33, DOI 10.1145/1496091.1496095, December 2008, <<https://doi.org/10.1145/1496091.1496095>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <<https://www.rfc-editor.org/info/rfc3022>>.
- [RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <<https://www.rfc-editor.org/info/rfc4787>>.
- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <<https://www.rfc-editor.org/info/rfc4963>>.
- [RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008, <<https://www.rfc-editor.org/info/rfc5382>>.
- [RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, DOI 10.17487/RFC5508, April 2009, <<https://www.rfc-editor.org/info/rfc5508>>.
- [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <<https://www.rfc-editor.org/info/rfc6056>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <<https://www.rfc-editor.org/info/rfc6269>>.
- [RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <<https://www.rfc-editor.org/info/rfc6333>>.
- [RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <<https://www.rfc-editor.org/info/rfc6458>>.
- [RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <<https://www.rfc-editor.org/info/rfc6890>>.
- [RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <<https://www.rfc-editor.org/info/rfc6951>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar, S., and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements", BCP 127, RFC 7857, DOI 10.17487/RFC7857, April 2016, <<https://www.rfc-editor.org/info/rfc7857>>.

- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <<https://www.rfc-editor.org/info/rfc8900>>.

Acknowledgments

The authors wish to thank Mohamed Boucadair, Gorrry Fairhurst, Bryan Ford, David Hayes, Alfred Hines, Karen E. E. Nielsen, Henning Peters, Maksim Proshin, Timo Völker, Dan Wing, and Qiaobing Xie for their invaluable comments.

In addition, the authors wish to thank David Hayes, Jason But, and Grenville Armitage, the authors of [DOI_10.1145_1496091.1496095], for their suggestions.

The authors also wish to thank Mohamed Boucadair for contributing the text related to the YANG module.

Authors' Addresses

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

Email: randall@lakerest.net

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

Email: tuexen@fh-muenster.de

Irene Rüngeler
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: i.ruengeler@fh-muenster.de

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 25, 2019

R. Stewart
Netflix, Inc.
M. Tuexen
Muenster Univ. of Appl. Sciences
M. Proshin
Ericsson
October 22, 2018

RFC 4960 Errata and Issues
draft-ietf-tsvwg-rfc4960-errata-08.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 ordered 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 <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 April 25, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions	4
3. Corrections to RFC 4960	4
3.1. Path Error Counter Threshold Handling	4
3.2. Upper Layer Protocol Shutdown Request Handling	5
3.3. Registration of New Chunk Types	6
3.4. Variable Parameters for INIT Chunks	7
3.5. CRC32c Sample Code on 64-bit Platforms	8
3.6. Endpoint Failure Detection	9
3.7. Data Transmission Rules	10
3.8. T1-Cookie Timer	11
3.9. Miscellaneous Typos	12
3.10. CRC32c Sample Code	19
3.11. partial_bytes_acked after T3-rtx Expiration	20
3.12. Order of Adjustments of partial_bytes_acked and cwnd	21
3.13. HEARTBEAT ACK and the association error counter	22
3.14. Path for Fast Retransmission	23
3.15. Transmittal in Fast Recovery	24
3.16. Initial Value of ssthresh	25
3.17. Automatically Confirmed Addresses	26
3.18. Only One Packet after Retransmission Timeout	27
3.19. INIT ACK Path for INIT in COOKIE-WAIT State	28
3.20. Zero Window Probing and Unreachable Primary Path	29
3.21. Normative Language in Section 10	30
3.22. Increase of partial_bytes_acked in Congestion Avoidance	33
3.23. Inconsistency in Notifications Handling	34
3.24. SACK.Delay Not Listed as a Protocol Parameter	40
3.25. Processing of Chunks in an Incoming SCTP Packet	42
3.26. CWND Increase in Congestion Avoidance Phase	43
3.27. Refresh of cwnd and ssthresh after Idle Period	46
3.28. Window Updates After Receiver Window Opens Up	47
3.29. Path of DATA and Reply Chunks	48
3.30. Outstanding Data, Flightsize and Data In Flight Key Terms	50
3.31. CWND Degradation due to Max.Burst	52
3.32. Reduction of RTO.Initial	53
3.33. Ordering of Bundled SACK and ERROR Chunks	55
3.34. Undefined Parameter Returned by RECEIVE Primitive	56
3.35. DSCP Changes	57

3.36. Inconsistent Handling of ICMPv4 and ICMPv6 Messages . . .	58
3.37. Handling of Soft Errors	60
3.38. Honoring CWND	60
3.39. Zero Window Probing	62
3.40. Updating References Regarding ECN	64
3.41. Host Name Address Parameter Deprecated	66
3.42. Conflicting Text Regarding the Supported Address Types Parameter	70
3.43. Integration of RFC 6096	71
3.44. Integration of RFC 6335	73
3.45. Integration of RFC 7053	75
3.46. CRC32c Code Improvements	79
3.47. Clarification of Gap Ack Blocks in SACK Chunks	89
3.48. Handling of SSN Wrap Arounds	91
3.49. Update RFC 2119 Boilerplate	92
3.50. Missed Text Removal	93
4. IANA Considerations	94
5. Security Considerations	94
6. Acknowledgments	94
7. References	95
7.1. Normative References	95
7.2. Informative References	95
Authors' Addresses	96

1. Introduction

This document contains a compilation of all defects found up until the publication of this document for [RFC4960] specifying the Stream Control Transmission Protocol (SCTP). These defects may be of an editorial or technical nature. This document may be thought of as a companion document to be used in the implementation of SCTP to clarify errors in the original SCTP document.

This document provides a history of the changes that will be compiled into a BIS document for [RFC4960]. It is structured similar to [RFC4460].

Each error will be detailed within this document in the form of:

- o The problem description,
- o The text quoted from [RFC4960],
- o The replacement text that should be placed into an upcoming BIS document,
- o A description of the solution.

Note that when reading this document one must use care to assure that a field or item is not updated further on within the document. 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, the last delta in the text is the one which should be applied.

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. Corrections to RFC 4960

[NOTE to RFC-Editor:

References to obsoleted RFCs are in OLD TEXT sections and have the corresponding references to the obsoleting RFCs in the NEW TEXT sections. In addition to this, there are some references to the obsoleted [RFC2960], which are intended.

]

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.

This text has been modified by multiple errata. It is further updated in Section 3.23.

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 ignore ULP shutdown requests, but MUST continue responding to SHUTDOWN chunks from its peer.

This text is in final form, and is not further updated in this document.

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 [RFC8126]. Documentation of the chunk type MUST contain the following information:

This text has been modified by multiple errata. It is further updated in Section 3.43.

3.3.3. Solution Description

Refer to chunk types as intended and change reference to [RFC8126].

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

This text is in final form, and is not further updated in this document.

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;
```

This text has been modified by multiple errata. It is further updated in Section 3.10 and in Section 3.46.

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 SHOULD 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 is currently used for data transfer. Unacknowledged HEARTBEAT chunks observed on paths different from the path currently used for data transfer SHOULD NOT increment the association error counter, as this could lead to association closure even if the path which is currently used for data transfer is available (but idle).

This text has been modified by multiple errata. It is further updated in Section 3.23.

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 [RFC1122]. The algorithm can be similar to the one described in Section 4.2.3.4 of [RFC1122].

This text is in final form, and is not further updated in this document.

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)

```

This text has been modified by multiple errata. It is further updated in Section 3.9.

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 caught.

One typo was reported as an Errata for [RFC4960] with Errata ID 5003.

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 $TSN = 0$.

This text is in final form, and is not further updated in this document.

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.

This text is in final form, and is not further updated in this document.

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,
                                   /
                                   /
                                   <-----/ (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,
                                   /
                                   /
                                   <-----/
(remove 6 from out-queue,
and mark 7 as "1" missing report)

```

This text is in final form, and is not further updated in this document.

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 than 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 PMTU.

This text is in final form, and is not further updated in this document.

Old text: (Section 10.1 O))

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))

O) Receive Unacknowledged Message

Format: RECEIVE_UNACKED(data retrieval id, buffer address, buffer size [,stream id] [,stream sequence number] [,partial flag] [,payload protocol-id])

This text is in final form, and is not further updated in this document.

Old text: (Section 10.1 M)

M) Set Protocol Parameters

Format: SETPROTOCOLPARAMETERS(association id,
[,destination transport address,]
protocol parameter list)

New text: (Section 10.1 M)

M) Set Protocol Parameters

Format: SETPROTOCOLPARAMETERS(association id,
[destination transport address,]
protocol parameter list)

This text is in final form, and is not further updated in this document.

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".

This text is in final form, and is not further updated in this document.

Old text: (Appendix C)

ICMP7) If the ICMP message is either a v6 "Packet Too Big" or a v4 "Fragmentation Needed", an implementation MAY process this information as defined for PATH MTU discovery.

New text: (Appendix C)

ICMP7) If the ICMP message is either a v6 "Packet Too Big" or a v4 "Fragmentation Needed", an implementation MAY process this information as defined for PMTU discovery.

This text is in final form, and is not further updated in this document.

Old text: (Section 5.4)

2) For the receiver of the COOKIE ECHO, the only CONFIRMED address is the one to which the INIT-ACK was sent.

New text: (Section 5.4)

2) For the receiver of the COOKIE ECHO, the only CONFIRMED address is the one to which the INIT ACK was sent.

This text is in final form, and is not further updated in this 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)

```

This text has been modified by multiple errata. It includes modifications from Section 3.8. It is in final form, and is not further updated in this document.

 Old text: (Section 5.2.5)

5.2.5. Handle Duplicate COOKIE-ACK.

 New text: (Section 5.2.5)

5.2.5. Handle Duplicate COOKIE ACK.

This text is in final form, and is not further updated in this document.

Old text: (Section 8.3)

By default, an SCTP endpoint SHOULD monitor the reachability of the idle destination transport address(es) of its peer by sending a HEARTBEAT chunk periodically to the destination transport address(es). HEARTBEAT sending MAY begin upon reaching the ESTABLISHED state and is discontinued after sending either SHUTDOWN or SHUTDOWN-ACK. A receiver of a HEARTBEAT MUST respond to a HEARTBEAT with a HEARTBEAT-ACK after entering the COOKIE-ECHOED state (INIT sender) or the ESTABLISHED state (INIT receiver), up until reaching the SHUTDOWN-SENT state (SHUTDOWN sender) or the SHUTDOWN-ACK-SENT state (SHUTDOWN receiver).

New text: (Section 8.3)

By default, an SCTP endpoint SHOULD monitor the reachability of the idle destination transport address(es) of its peer by sending a HEARTBEAT chunk periodically to the destination transport address(es). HEARTBEAT sending MAY begin upon reaching the ESTABLISHED state and is discontinued after sending either SHUTDOWN or SHUTDOWN ACK. A receiver of a HEARTBEAT MUST respond to a HEARTBEAT with a HEARTBEAT ACK after entering the COOKIE-ECHOED state (INIT sender) or the ESTABLISHED state (INIT receiver), up until reaching the SHUTDOWN-SENT state (SHUTDOWN sender) or the SHUTDOWN-ACK-SENT state (SHUTDOWN receiver).

This text is in final form, and is not further updated in this document.

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 all of Appendix C starting with the following sentence to the end of Appendix B.

The following non-normative sample code is taken from an open-source CRC generator [WILLIAMS93], using the "mirroring" technique and yielding a lookup table for SCTP CRC32c with 256 entries, each 32 bits wide.

This text has been modified by multiple errata. It includes modifications from Section 3.5. It is further updated in Section 3.46.

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 detection 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
```

This text is in final form, and is not further updated in this document.

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] likely implies the wrong 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 `1*MTU`.

This text has been modified by multiple errata. It is further updated in Section 3.26.

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 MUST 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.

This text is in final form, and is not further updated in this document.

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 MUST 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).

This text has been modified by multiple errata. It is further updated in Section 3.23.

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 the original data was sent to. If the primary path has been changed and the original data was sent to the old primary path before the fast retransmit, the implementation MAY send it to the new primary path.

This text is in final form, and is not further updated in this document.

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 PMTU 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.

This text is in final form, and is not further updated in this document.

3.15.3. Solution Description

The new text explicitly specifies to send only the first packet in the Fast Recovery phase disregarding cwnd limitations.

3.16. Initial Value of ssthresh

3.16.1. Description of the Problem

The initial value of ssthresh should be set arbitrarily high. Using the advertised receiver window of the peer is inappropriate if the peer increases its window after the handshake. Furthermore, use a higher requirements level, since not following the advice may result in performance problems.

3.16.2. Text Changes to the Document

Old text: (Section 7.2.1)

- o The initial value of ssthresh MAY be arbitrarily high (for example, implementations MAY use the size of the receiver advertised window).

New text: (Section 7.2.1)

- o The initial value of ssthresh SHOULD be arbitrarily high (e.g., the size of the largest possible advertised window).

This text is in final form, and is not further updated in this document.

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 are automatically considered to be CONFIRMED.

This text is in final form, and is not further updated in this document.

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.

This text is in final form, and is not further updated in this document.

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.

This text is in final form, and is not further updated in this document.

3.19.3. Solution Description

The new text requires sending INIT ACK to a destination address that is passed by the upper layer and reported in the incoming INIT. If the source address of the INIT meets these conditions, 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. However, the reception of new packets from the peer does not guarantee the peer's reachability and, if the destination address becomes unreachable during zero window probing,

SCTP cannot get an updated 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 could keep its window closed for an indefinite time. Section 6.2 describes the receiver behavior when it advertises a zero window.

This text is in final form, and is not further updated in this document.

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, therefore, 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))

- 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))

- 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.

This text is in final form, and is not further updated in this document.

Old text: (Section 10.1 G))

- o Stream Sequence Number - the Stream Sequence Number assigned by the sending SCTP peer.
- 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))

- o stream sequence number - the Stream Sequence Number assigned by the sending SCTP peer.
- o partial flag - if this returned flag is set to 1, then this primitive contains a partial delivery of the whole message. When this flag is set, the stream id and stream sequence number must accompany this primitive. When this flag is set to 0, it indicates that no more deliveries will be received for this stream sequence number.

This text is in final form, and is not further updated in this document.

Old text: (Section 10.1 N)

- o Stream Sequence Number - this value is returned indicating the Stream Sequence Number that was associated with the message.
- 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)

- o stream sequence number - this value is returned indicating the Stream Sequence Number that was associated with the message.
- 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 primitive. When this flag is set to 0, it indicates that no more deliveries will be received for this stream sequence number.

This text is in final form, and is not further updated in this document.

Old text: (Section 10.1 O)

- o Stream Sequence Number - this value is returned indicating the Stream Sequence Number that was associated with the message.
- 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)

- o stream sequence number - this value is returned indicating the Stream Sequence Number that was associated with the message.
- 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 primitive. When this flag is set to 0, it indicates that no more deliveries will be received for this stream sequence number.

This text is in final form, and is not further updated in this document.

3.21.3. Solution Description

The normative language is removed from Section 10. In addition, the consistency of the text has been improved.

3.22. Increase of partial_bytes_acked in Congestion Avoidance

3.22.1. Description of the Problem

Two issues have been discovered with the partial_bytes_acked handling described in Section 7.2.2 of [RFC4960]:

- o If the Cumulative TSN Ack Point is not advanced but the SACK chunk acknowledges new TSNs in the Gap Ack Blocks, these newly acknowledged TSNs are not considered for partial_bytes_acked although these TSNs were successfully received by the peer.

- o Duplicate TSNs are not considered in `partial_bytes_acked` although they confirm that the DATA chunks were successfully received by the peer.

3.22.2. Text Changes to the Document

Old text: (Section 7.2.2)

- o Whenever `cwnd` is greater than `ssthresh`, upon each SACK arrival that advances the Cumulative TSN Ack Point, increase `partial_bytes_acked` by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack and by Gap Ack Blocks.

New text: (Section 7.2.2)

- o Whenever `cwnd` is greater than `ssthresh`, upon each SACK arrival, increase `partial_bytes_acked` by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack, by Gap Ack Blocks and by the number of bytes of duplicated chunks reported in Duplicate TSNs.

This text has been modified by multiple errata. It is further updated in Section 3.26.

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

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 SHOULD 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 SHOULD 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 SHOULD 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.

This text has been modified by multiple errata. It includes modifications from Section 3.6. It is in final form, and is not further updated in this document.

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 SHOULD 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 could 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.

This text is in final form, and is not further updated in this 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 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.

This text has been modified by multiple errata. It includes modifications from Section 3.1. It is in final form, and is not further updated in this document.

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).

This text has been modified by multiple errata. It includes modifications from Section 3.13. It is in final form, and is not further updated in this document.

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).

This text is in final form, and is not further updated in this document.

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).

This text is in final form, and is not further updated in this document.

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.

This text is in final form, and is not further updated in this document.

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

This text has been modified by multiple errata. It is further updated in Section 3.32.

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 not to perform any prescreening of chunks in the received packet. Thus, by discarding one chunk the receiver also causes discarding of 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'.

This text is in final form, and is not further updated in this document.

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. The receiver of an INIT chunk MUST silently discard 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.

This text is in final form, and is not further updated in this document.

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 data outstanding 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

this but this algorithm is underspecified and actually allows increasing cwnd by more than 1*MTU per RTT.

3.26.2. Text Changes to the Document

Old text: (Section 7.2.2)

When cwnd is greater than ssthresh, cwnd should be incremented by 1*MTU per RTT if the sender has cwnd or more bytes of data outstanding for the corresponding transport address.

New text: (Section 7.2.2)

When cwnd is greater than ssthresh, cwnd SHOULD be incremented by 1*MTU per RTT if the sender has cwnd or more bytes of data outstanding for the corresponding transport address. The basic guidelines for incrementing cwnd during congestion avoidance are:

- o Sctp MAY increment cwnd by 1*MTU.
- o Sctp SHOULD increment cwnd by one 1*MTU once per RTT when the sender has cwnd or more bytes of data outstanding for the corresponding transport address.
- o Sctp MUST NOT increment cwnd by more than 1*MTU per RTT.

This text is in final form, and is not further updated in this document.

Old text: (Section 7.2.2)

- o Whenever cwnd is greater than ssthresh, upon each SACK arrival that advances the Cumulative TSN Ack Point, increase partial_bytes_acked by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack and by Gap Ack Blocks.
- o When partial_bytes_acked is equal to or greater than cwnd and before the arrival of the SACK the sender had cwnd or more bytes of data outstanding (i.e., before arrival of the SACK, flightsize was greater than or equal to cwnd), increase cwnd by MTU, and reset partial_bytes_acked to (partial_bytes_acked - cwnd).

New text: (Section 7.2.2)

- o Whenever cwnd is greater than ssthresh, upon each SACK arrival, increase partial_bytes_acked by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack, by Gap Ack Blocks and by the number of bytes of duplicated chunks reported in Duplicate TSNs.
- o When partial_bytes_acked is greater than cwnd and before the arrival of the SACK the sender had less than cwnd bytes of data outstanding (i.e., before arrival of the SACK, flightsize was less than cwnd), reset partial_bytes_acked to cwnd.
- o When partial_bytes_acked is equal to or greater than cwnd and before the arrival of the SACK the sender had cwnd or more bytes of data outstanding (i.e., before arrival of the SACK, flightsize was greater than or equal to cwnd), partial_bytes_acked is reset to (partial_bytes_acked - cwnd). Next, cwnd is increased by 1*MTU.

This text has been modified by multiple errata. It includes modifications from Section 3.12 and Section 3.22. It is in final form, and is not further updated in this document.

3.26.3. Solution Description

The basic guidelines for incrementing cwnd during the 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 ssthresh 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 that switches back to the previously failed primary path will also lead to the situation where traffic ends up in Congestion Avoidance.

3.27.2. Text Changes to the Document

Old text: (Section 7.2.1)

- o The initial cwnd before DATA transmission or after a sufficiently long idle period MUST be set to min(4*MTU, max (2*MTU, 4380 bytes)).

New text: (Section 7.2.1)

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

Old text: (Section 7.2.1)

- o When the endpoint does not transmit data on a given transport address, the cwnd of the transport address should be adjusted to $\max(\text{cwnd}/2, 4 \cdot \text{MTU})$ per RTO.

New text: (Section 7.2.1)

- o While the endpoint does not transmit data on a given transport address, the cwnd of the transport address SHOULD be adjusted to $\max(\text{cwnd}/2, 4 \cdot \text{MTU})$ once per RTO. Before the first cwnd adjustment, the ssthresh of the transport address SHOULD be set to the cwnd.

This text is in final form, and is not further updated in this document.

3.27.3. Solution Description

A rule about cwnd adjustment after a sufficiently long idle period is removed.

The text is updated to describe the ssthresh handling. 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 a large number of window updates, in particular large bursts of them. One way to achieve this is to send a window update only if the window can be increased by at least a quarter of the receive buffer size of the association.

This text is in final form, and is not further updated in this document.

3.28.3. Solution Description

The new text makes clear that additional SACK chunks for window updates should be sent as long as excessive bursts are avoided.

3.29. Path of DATA and Reply Chunks

3.29.1. Description of the Problem

Section 6.4 of [RFC4960] describes the transmission policy for multi-homed SCTP endpoints. However, there are the following issues with it:

- o It states that a SACK should be sent to the source address of an incoming DATA. However, it is known that other SACK policies (e.g. sending SACKs always to the primary path) may be more beneficial in some situations.
- o Initially it states that an endpoint should always transmit DATA chunks to the primary path. Then it states that the rule for transmittal of reply chunks should also be followed if the endpoint is bundling DATA chunks together with the reply chunk which contradicts with the first statement to always transmit DATA

chunks to the primary path. Some implementations were having problems with it and sent DATA chunks bundled with reply chunks to a different destination address than the primary path that caused many gaps.

3.29.2. Text Changes to the Document

Old text: (Section 6.4)

An endpoint SHOULD transmit reply chunks (e.g., SACK, HEARTBEAT ACK, etc.) to the same destination transport address from which it received the DATA or control chunk to which it is replying. This rule should also be followed if the endpoint is bundling DATA chunks together with the reply chunk.

However, when acknowledging multiple DATA chunks received in packets from different source addresses in a single SACK, the SACK chunk may be transmitted to one of the destination transport addresses from which the DATA or control chunks being acknowledged were received.

New text: (Section 6.4)

An endpoint SHOULD transmit reply chunks (e.g., INIT ACK, COOKIE ACK, HEARTBEAT ACK, etc.) in response to control chunks to the same destination transport address from which it received the control chunk to which it is replying.

The selection of the destination transport address for packets containing SACK chunks is implementation dependent. However, an endpoint SHOULD NOT vary the destination transport address of a SACK when it receives DATA chunks coming 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.

This text is in final form, and is not further updated in this document.

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.

3.30. Outstanding Data, Flightsize and Data In Flight Key Terms

3.30.1. Description of the Problem

[RFC4960] uses outstanding data, flightsize and data in flight key terms in formulas and statements but their definitions are not provided in Section 1.3. Furthermore, outstanding data does not include DATA chunks which are classified as lost but which have not been retransmitted yet and there is a paragraph in Section 6.1 of [RFC4960] where this statement is broken.

3.30.2. Text Changes to the Document

Old text: (Section 1.3)

- o Congestion window (cwnd): An SCTP variable that limits the data, in number of bytes, a sender can send to a particular destination transport address before receiving an acknowledgement.

...

- o Outstanding TSN (at an SCTP endpoint): A TSN (and the associated DATA chunk) that has been sent by the endpoint but for which it has not yet received an acknowledgement.

New text: (Section 1.3)

- o Outstanding TSN (at an SCTP endpoint): A TSN (and the associated DATA chunk) that has been sent by the endpoint but for which it has not yet received an acknowledgement.
- o Outstanding data (or Data outstanding or Data in flight): The total amount of the DATA chunks associated with outstanding TSNs. A retransmitted DATA chunk is counted once in outstanding data. A DATA chunk which is classified as lost but which has not been retransmitted yet is not in outstanding data.
- o Flightsize: The amount of bytes of outstanding data to a particular destination transport address at any given time.
- o Congestion window (cwnd): An SCTP variable that limits outstanding data, in number of bytes, a sender can send to a particular destination transport address before receiving an acknowledgement.

This text is in final form, and is not further updated in this document.

Old text: (Section 6.1)

- C) When the time comes for the sender to transmit, before sending new DATA chunks, the sender MUST first transmit any outstanding DATA chunks that are marked for retransmission (limited by the current cwnd).

New text: (Section 6.1)

- C) When the time comes for the sender to transmit, before sending new DATA chunks, the sender MUST first transmit any DATA chunks that are marked for retransmission (limited by the current cwnd).

This text is in final form, and is not further updated in this document.

3.30.3. Solution Description

Now Section 1.3, Key Terms, includes explanations of outstanding data, data in flight and flightsize key terms. Section 6.1 is corrected to properly use the outstanding data term.

3.31. CWND Degradation due to Max.Burst

3.31.1. Description of the Problem

Some implementations were experiencing a degradation of cwnd because of the Max.Burst limit. This was due to misinterpretation of the suggestion in [RFC4960], Section 6.1, on how to use the Max.Burst parameter when calculating the number of packets to transmit.

3.31.2. Text Changes to the Document

Old text: (Section 6.1)

- D) When the time comes for the sender to transmit new DATA chunks, the protocol parameter Max.Burst SHOULD be used to limit the number of packets sent. The limit MAY be applied by adjusting cwnd as follows:

```
if((flightsize + Max.Burst*MTU) < cwnd) cwnd = flightsize +
Max.Burst*MTU
```

Or it MAY be applied by strictly limiting the number of packets emitted by the output routine.

New text: (Section 6.1)

- D) When the time comes for the sender to transmit new DATA chunks, the protocol parameter Max.Burst SHOULD be used to limit the number of packets sent. The limit MAY be applied by adjusting cwnd temporarily as follows:

```
if ((flightsize + Max.Burst*MTU) < cwnd)
    cwnd = flightsize + Max.Burst*MTU
```

Or it MAY be applied by strictly limiting the number of packets emitted by the output routine. When calculating the number of packets to transmit and particularly using the formula above, cwnd SHOULD NOT be changed permanently.

This text is in final form, and is not further updated in this document.

3.31.3. Solution Description

The new text clarifies that cwnd should not be changed when applying the Max.Burst limit. This mitigates packet bursts related to the reception of SACK chunks, but not bursts related to an application sending a burst of user messages.

3.32. Reduction of RTO.Initial

3.32.1. Description of the Problem

[RFC4960] uses 3 seconds as the default value for RTO.Initial in accordance with Section 4.3.2.1 of [RFC1122]. [RFC6298] updates [RFC1122] and lowers the initial value of the retransmission timer from 3 seconds to 1 second.

3.32.2. Text Changes to the Document

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 - 1 second
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

This text has been modified by multiple errata. It includes modifications from Section 3.24. It is in final form, and is not further updated in this document.

3.32.3. Solution Description

The value `RTO.Initial` has been lowered to 1 second to be in tune with [RFC6298].

3.33. Ordering of Bundled SACK and ERROR Chunks

3.33.1. Description of the Problem

When an SCTP endpoint receives a DATA chunk with an invalid stream identifier it shall acknowledge it by sending a SACK chunk and indicate that the stream identifier was invalid by sending an ERROR chunk. These two chunks may be bundled. However, [RFC4960] requires in case of bundling that the ERROR chunk follows the SACK chunk. This restriction of the ordering is not necessary and might only limit interoperability.

3.33.2. Text Changes to the Document

Old text: (Section 6.5)

Every DATA chunk MUST carry a valid stream identifier. If an endpoint receives a DATA chunk with an invalid stream identifier, it shall acknowledge the reception of the DATA chunk following the normal procedure, immediately send an ERROR chunk with cause set to "Invalid Stream Identifier" (see Section 3.3.10), and discard the DATA chunk. The endpoint may bundle the ERROR chunk in the same packet as the SACK as long as the ERROR follows the SACK.

New text: (Section 6.5)

Every DATA chunk MUST carry a valid stream identifier. If an endpoint receives a DATA chunk with an invalid stream identifier, it SHOULD acknowledge the reception of the DATA chunk following the normal procedure, immediately send an ERROR chunk with cause set to "Invalid Stream Identifier" (see Section 3.3.10), and discard the DATA chunk. The endpoint MAY bundle the ERROR chunk and the SACK Chunk in the same packet.

This text is in final form, and is not further updated in this document.

3.33.3. Solution Description

The unnecessary restriction regarding the ordering of the SACK and ERROR chunk has been removed.

3.34. Undefined Parameter Returned by RECEIVE Primitive

3.34.1. Description of the Problem

[RFC4960] provides a description of an abstract API. In the definition of the RECEIVE primitive an optional parameter with name "delivery number" is mentioned. However, no definition of this parameter is given in [RFC4960] and the parameter is unnecessary.

3.34.2. Text Changes to the Document

Old text: (Section 10.1 G))

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]

New text: (Section 10.1 G))

G) Receive

Format: RECEIVE(association id, buffer address, buffer size
[,stream id])

-> byte count [,transport address] [,stream id] [,stream sequence
number] [,partial flag] [,payload protocol-id]

This text is in final form, and is not further updated in this document.

3.34.3. Solution Description

The undefined parameter has been removed.

3.35. DSCP Changes

3.35.1. Description of the Problem

The upper layer can change the Differentiated Services Code Point (DSCP) used for packets being sent. A change of the DSCP can result in packets hitting different queues on the path and, therefore, the congestion control should be initialized when the DSCP is changed by the upper layer. This is not described in [RFC4960].

3.35.2. Text Changes to the Document

New text: (Section 7.2.5)

7.2.5. Change of Differentiated Services Code Points

SCTP implementations MAY allow an application to configure the Differentiated Services Code Point (DSCP) used for sending packets. If a DSCP change might result in outgoing packets being queued in different queues, the congestion control parameters for all affected destination addresses MUST be reset to their initial values.

This text is in final form, and is not further updated in this document.

Old text: (Section 10.1 M)

Mandatory attributes:

- o association id - local handle to the SCTP association.
- o protocol parameter list - the specific names and values of the protocol parameters (e.g., Association.Max.Retrans; see Section 15) that the SCTP user wishes to customize.

New text: (Section 10.1 M)

Mandatory attributes:

- o association id - local handle to the SCTP association.
- o protocol parameter list - the specific names and values of the protocol parameters (e.g., Association.Max.Retrans; see Section 15, or other parameters like the DSCP) that the SCTP user wishes to customize.

This text is in final form, and is not further updated in this document.

3.35.3. Solution Description

Text describing the required action on DSCP changes has been added.

3.36. Inconsistent Handling of ICMPv4 and ICMPv6 Messages

3.36.1. Description of the Problem

Appendix C of [RFC4960] describes the handling of ICMPv4 and ICMPv6 messages. The handling of ICMP messages indicating that the port number is unreachable described in the enumeration is not consistent with the description given in [RFC4960] after the enumeration. Furthermore, the text explicitly describes the handling of ICMPv6 packets indicating reachability problems, but does not do the same for the corresponding ICMPv4 packets.

3.36.2. Text Changes to the Document

Old text: (Appendix C)

ICMP3) An implementation MAY ignore any ICMPv4 messages where the code does not indicate "Protocol Unreachable" or "Fragmentation Needed".

New text: (Appendix C)

ICMP3) An implementation SHOULD ignore any ICMP messages where the code indicates "Port Unreachable".

This text is in final form, and is not further updated in this document.

Old text: (Appendix C)

ICMP9) If the ICMPv6 code is "Destination Unreachable", the implementation MAY mark the destination into the unreachable state or alternatively increment the path error counter.

New text: (Appendix C)

ICMP9) If the ICMP type is "Destination Unreachable", the implementation MAY mark the destination into the unreachable state or alternatively increment the path error counter.

This text has been modified by multiple errata. It is further updated in Section 3.37.

3.36.3. Solution Description

The text has been changed to describe the intended handling of ICMP messages indicating that the port number is unreachable by replacing the third rule. Furthermore, remove the limitation to ICMPv6 in the ninth rule.

3.37. Handling of Soft Errors

3.37.1. Description of the Problem

[RFC1122] defines the handling of soft errors and hard errors for TCP. Appendix C of [RFC4960] only deals with hard errors.

3.37.2. Text Changes to the Document

Old text: (Appendix C)

ICMP9) If the ICMPv6 code is "Destination Unreachable", the implementation MAY mark the destination into the unreachable state or alternatively increment the path error counter.

New text: (Appendix C)

ICMP9) If the ICMP type is "Destination Unreachable", the implementation MAY mark the destination into the unreachable state or alternatively increment the path error counter. SCTP MAY provide information to the upper layer indicating the reception of ICMP messages when reporting a network status change.

This text has been modified by multiple errata. It includes modifications from Section 3.36. It is in final form, and is not further updated in this document.

3.37.3. Solution Description

Text has been added allowing SCTP to notify the application in case of soft errors.

3.38. Honoring CWND

3.38.1. Description of the Problem

When using the slow start algorithm, SCTP increases the congestion window only when it is being fully utilized. Since SCTP uses DATA chunks and does not use the congestion window to fragment user messages, this requires that some overbooking of the congestion window is allowed.

3.38.2. Text Changes to the Document

Old text: (Section 6.1)

- B) At any given time, the sender MUST NOT transmit new data to a given transport address if it has cwnd or more bytes of data outstanding to that transport address.

New text: (Section 6.1)

- B) At any given time, the sender MUST NOT transmit new data to a given transport address if it has cwnd + (PMTU - 1) or more bytes of data outstanding to that transport address. If data is available the sender SHOULD exceed cwnd by up to (PMTU-1) bytes on a new data transmission if the flightsize does not currently reach cwnd. The breach of cwnd MUST constitute one packet only.

This text is in final form, and is not further updated in this document.

Old text: (Section 7.2.1)

- o Whenever cwnd is greater than zero, the endpoint is allowed to have cwnd bytes of data outstanding on that transport address.

New text: (Section 7.2.1)

- o Whenever cwnd is greater than zero, the endpoint is allowed to have cwnd bytes of data outstanding on that transport address. A limited overbooking as described in B) of Section 6.1 SHOULD be supported.

This text is in final form, and is not further updated in this document.

3.38.3. Solution Description

Text was added to clarify how the cwnd limit should be handled.

3.39. Zero Window Probing

3.39.1. Description of the Problem

The text describing zero window probing was not clearly handling the case where the window was not zero, but too small for the next DATA chunk to be transmitted. Even in this case, zero window probing has to be performed to avoid deadlocks.

3.39.2. Text Changes to the Document

Old text: (Section 6.1)

- A) At any given time, the data sender MUST NOT transmit new data to any destination transport address if its peer's rwnd indicates that the peer has no buffer space (i.e., rwnd is 0; see Section 6.2.1). 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's having been lost in transit from the data receiver to the data sender.

When the receiver's advertised window is zero, this probe is called a zero window probe. Note that a zero window probe SHOULD only be sent when all outstanding DATA chunks have been cumulatively acknowledged and no DATA chunks are in flight. Zero window probing MUST be supported.

New text: (Section 6.1)

- A) At any given time, the data sender MUST NOT transmit new data to any destination transport address if its peer's rwnd indicates that the peer has no buffer space (i.e., rwnd is smaller than the size of the next DATA chunk; see Section 6.2.1). 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's having been lost in transit from the data receiver to the data sender.

When the receiver has no buffer space, this probe is called a zero window probe. Note that a zero window probe SHOULD only be sent when all outstanding DATA chunks have been cumulatively acknowledged and no DATA chunks are in flight. Zero window probing MUST be supported.

This text is in final form, and is not further updated in this document.

3.39.3. Solution Description

The terminology is used in a cleaner way.

3.40. Updating References Regarding ECN

3.40.1. Description of the Problem

[RFC4960] refers for ECN only to [RFC3168], which will be updated by [RFC8311]. This needs to be reflected when referring to ECN.

3.40.2. Text Changes to the Document

Old text: (Appendix A)

ECN [RFC3168] describes a proposed extension to IP that details a method to become aware of congestion outside of datagram loss.

New text: (Appendix A)

ECN as specified in [RFC3168] updated by [RFC8311] describes an extension to IP that details a method to become aware of congestion outside of datagram loss.

This text is in final form, and is not further updated in this document.

Old text: (Appendix A)

In general, [RFC3168] should be followed with the following exceptions.

New text: (Appendix A)

In general, [RFC3168] updated by [RFC8311] SHOULD be followed with the following exceptions.

This text is in final form, and is not further updated in this document.

Old text: (Appendix A)

[RFC3168] details negotiation of ECN during the SYN and SYN-ACK stages of a TCP connection.

New text: (Appendix A)

[RFC3168] updated by [RFC8311] details negotiation of ECN during the SYN and SYN-ACK stages of a TCP connection.

This text is in final form, and is not further updated in this document.

Old text: (Appendix A)

[RFC3168] details a specific bit for a receiver to send back in its TCP acknowledgements to notify the sender of the Congestion Experienced (CE) bit having arrived from the network.

New text: (Appendix A)

[RFC3168] updated by [RFC8311] details a specific bit for a receiver to send back in its TCP acknowledgements to notify the sender of the Congestion Experienced (CE) bit having arrived from the network.

This text is in final form, and is not further updated in this document.

Old text: (Appendix A)

[RFC3168] details a specific bit for a sender to send in the header of its next outbound TCP segment to indicate to its peer that it has reduced its congestion window.

New text: (Appendix A)

[RFC3168] updated by [RFC8311] details a specific bit for a sender to send in the header of its next outbound TCP segment to indicate to its peer that it has reduced its congestion window.

This text is in final form, and is not further updated in this document.

3.40.3. Solution Description

References to [RFC8311] have been added. While there, some wordsmithing has been performed.

3.41. Host Name Address Parameter Deprecated

3.41.1. Description of the Problem

[RFC4960] defines three types of address parameters to be used with INIT and INIT ACK chunks:

1. IPv4 Address parameters.
2. IPv6 Address parameters.
3. Host Name Address parameters.

The first two are supported by the SCTP kernel implementations of FreeBSD, Linux and Solaris, but the third one is not. In addition, the first two were successfully tested in all nine interoperability tests for SCTP, but the third one has never been successfully tested. Therefore, the Host Name Address parameter should be deprecated.

3.41.2. Text Changes to the Document

Old text: (Section 3.3.2)

Note 3: An INIT chunk MUST NOT contain more than one Host Name Address parameter. Moreover, the sender of the INIT MUST NOT combine any other address types with the Host Name Address in the INIT. The receiver of INIT MUST ignore any other address types if the Host Name Address parameter is present in the received INIT chunk.

New text: (Section 3.3.2)

Note 3: An INIT chunk MUST NOT contain the Host Name Address parameter. The receiver of an INIT chunk containing a Host Name Address parameter MUST send an ABORT and MAY include an Error Cause indicating an Unresolvable Address.

This text is in final form, and is not further updated in this document.

Old text: (Section 3.3.2.1)

The sender of INIT uses this parameter to pass its Host Name (in place of its IP addresses) to its peer. The peer is responsible for resolving the name. Using this parameter might make it more likely for the association to work across a NAT box.

New text: (Section 3.3.2.1)

The sender of an INIT chunk MUST NOT include this parameter. The usage of the Host Name Address parameter is deprecated.

This text is in final form, and is not further updated in this document.

Old text: (Section 3.3.2.1)

Address Type: 16 bits (unsigned integer)

This is filled with the type value of the corresponding address TLV (e.g., IPv4 = 5, IPv6 = 6, Host name = 11).

New text: (Section 3.3.2.1)

Address Type: 16 bits (unsigned integer)

This is filled with the type value of the corresponding address TLV (e.g., IPv4 = 5, IPv6 = 6). The value indicating the Host Name Address parameter (Host name = 11) MUST NOT be used.

This text is in final form, and is not further updated in this document.

Old text: (Section 3.3.3)

Note 3: The INIT ACK chunks MUST NOT contain more than one Host Name Address parameter. Moreover, the sender of the INIT ACK MUST NOT combine any other address types with the Host Name Address in the INIT ACK. The receiver of the INIT ACK MUST ignore any other address types if the Host Name Address parameter is present.

New text: (Section 3.3.3)

Note 3: An INIT ACK chunk MUST NOT contain the Host Name Address parameter. The receiver of INIT ACK chunks containing a Host Name Address parameter MUST send an ABORT and MAY include an Error Cause indicating an Unresolvable Address.

This text is in final form, and is not further updated in this document.

Old text: (Section 5.1.2)

B) If there is a Host Name parameter present in the received INIT or

INIT ACK chunk, the endpoint shall resolve that host name to a list of IP address(es) and derive the transport address(es) of this peer by combining the resolved IP address(es) with the SCTP source port.

The endpoint MUST ignore any other IP Address parameters if they are also present in the received INIT or INIT ACK chunk.

The time at which the receiver of an INIT resolves the host name has potential security implications to SCTP. If the receiver of an INIT resolves the host name upon the reception of the chunk, and the mechanism the receiver uses to resolve the host name involves potential long delay (e.g., DNS query), the receiver may open itself up to resource attacks for the period of time while it is waiting for the name resolution results before it can build the State Cookie and release local resources.

Therefore, in cases where the name translation involves potential long delay, the receiver of the INIT MUST postpone the name resolution till the reception of the COOKIE ECHO chunk from the peer. In such a case, the receiver of the INIT SHOULD build the State Cookie using the received Host Name (instead of destination transport addresses) and send the INIT ACK to the source IP address from which the INIT was received.

The receiver of an INIT ACK shall always immediately attempt to resolve the name upon the reception of the chunk.

The receiver of the INIT or INIT ACK MUST NOT send user data (piggy-backed or stand-alone) to its peer until the host name is successfully resolved.

If the name resolution is not successful, the endpoint MUST immediately send an ABORT with "Unresolvable Address" error cause to its peer. The ABORT shall be sent to the source IP address from which the last peer packet was received.

New text: (Section 5.1.2)

- B) If there is a Host Name parameter present in the received INIT or INIT ACK chunk, the endpoint MUST immediately send an ABORT and MAY include an Error Cause indicating an Unresolvable Address to its peer. The ABORT SHALL be sent to the source IP address from which the last peer packet was received.

This text is in final form, and is not further updated in this document.

Old text: (Section 11.2.4.1)

The use of the host name feature in the INIT chunk could be used to flood a target DNS server. A large backlog of DNS queries, resolving the host name received in the INIT chunk to IP addresses, could be accomplished by sending INITs to multiple hosts in a given domain. In addition, an attacker could use the host name feature in an indirect attack on a third party by sending large numbers of INITs to random hosts containing the host name of the target. In addition to the strain on DNS resources, this could also result in large numbers of INIT ACKs being sent to the target. One method to protect against this type of attack is to verify that the IP addresses received from DNS include the source IP address of the original INIT. If the list of IP addresses received from DNS does not include the source IP address of the INIT, the endpoint MAY silently discard the INIT. This last option will not protect against the attack against the DNS.

New text: (Section 11.2.4.1)

The support of the Host Name Address parameter has been removed from the protocol. Endpoints receiving INIT or INIT ACK chunks containing the Host Name Address parameter MUST send an ABORT chunk in response and MAY include an Error Cause indicating an Unresolvable Address.

This text is in final form, and is not further updated in this document.

3.41.3. Solution Description

The usage of the Host Name Address parameter has been deprecated.

3.42. Conflicting Text Regarding the Supported Address Types Parameter

3.42.1. Description of the Problem

When receiving an SCTP packet containing an INIT chunk sent from an address for which the corresponding address type is not listed in the Supported Address Types, there is conflicting text in Section 5.1.2 of [RFC4960]. It is stated that the association MUST be aborted and also that the association SHOULD be established and there SHOULD NOT be any error indication.

3.42.2. Text Changes to the Document

Old text: (Section 5.1.2)

The sender of INIT may include a 'Supported Address Types' parameter in the INIT to indicate what types of address are acceptable. When this parameter is present, the receiver of INIT (initiate) MUST either use one of the address types indicated in the Supported Address Types parameter when responding to the INIT, or abort the association with an "Unresolvable Address" error cause if it is unwilling or incapable of using any of the address types indicated by its peer.

New text: (Section 5.1.2)

The sender of INIT chunks MAY include a 'Supported Address Types' parameter in the INIT to indicate what types of addresses are acceptable.

This text is in final form, and is not further updated in this document.

3.42.3. Solution Description

The conflicting text has been removed.

3.43. Integration of RFC 6096

3.43.1. Description of the Problem

[RFC6096] updates [RFC4960] by adding a Chunk Flags Registry. This should be integrated into the base specification.

3.43.2. Text Changes to the Document

Old text: (Section 14.1)

14.1. IETF-Defined Chunk Extension

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:

- a) A long and short name for the new chunk type.
- b) A detailed description of the structure of the chunk, which MUST conform to the basic structure defined in Section 3.2.
- c) A detailed definition and description of the intended use of each field within the chunk, including the chunk flags if any.
- d) A detailed procedural description of the use of the new chunk type within the operation of the protocol.

The last chunk type (255) is reserved for future extension if necessary.

New text: (Section 14.1)

14.1. IETF-Defined Chunk Extension

The assignment of new chunk type codes is done through an IETF Review action, as defined in [RFC8126]. Documentation of a new chunk MUST contain the following information:

- a) A long and short name for the new chunk type;
- b) A detailed description of the structure of the chunk, which MUST conform to the basic structure defined in Section 3.2 of [RFC4960];
- c) A detailed definition and description of the intended use of each field within the chunk, including the chunk flags if any. Defined chunk flags will be used as initial entries in the chunk flags table for the new chunk type;
- d) A detailed procedural description of the use of the new chunk type within the operation of the protocol.

The last chunk type (255) is reserved for future extension if necessary.

For each new chunk type, IANA creates a registration table for the chunk flags of that type. The procedure for registering particular chunk flags is described in the following Section 14.2.

This text has been modified by multiple errata. It includes modifications from Section 3.3. It is in final form, and is not further updated in this document.

New text: (Section 14.2)

14.2. New IETF Chunk Flags Registration

The assignment of new chunk flags is done through an RFC required action, as defined in [RFC8126]. Documentation of the chunk flags MUST contain the following information:

- a) A name for the new chunk flag;
- b) A detailed procedural description of the use of the new chunk flag within the operation of the protocol. It MUST be considered that implementations not supporting the flag will send '0' on transmit and just ignore it on receipt.

IANA selects a chunk flags value. This MUST be one of 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, or 0x80, which MUST be unique within the chunk flag values for the specific chunk type.

This text is in final form, and is not further updated in this document.

Please note that Sections 14.2, 14.3, 14.4, and 14.5 need to be renumbered.

3.43.3. Solution Description

[RFC6096] was integrated and the reference updated to [RFC8126].

3.44. Integration of RFC 6335

3.44.1. Description of the Problem

[RFC6335] updates [RFC4960] by updating Procedures for the Port Numbers Registry. This should be integrated into the base specification. While there, update the reference to the RFC giving guidelines for writing IANA sections to [RFC8126].

3.44.2. Text Changes to the Document

Old text: (Section 14.5)

SCTP services may use contact port numbers to provide service to unknown callers, as in TCP and UDP. IANA is therefore requested to

open the existing Port Numbers registry for SCTP using the following rules, which we intend to mesh well with existing Port Numbers registration procedures. An IESG-appointed Expert Reviewer supports IANA in evaluating SCTP port allocation requests, according to the procedure defined in [RFC2434].

Port numbers are divided into three ranges. The Well Known Ports are those from 0 through 1023, the Registered Ports are those from 1024 through 49151, and the Dynamic and/or Private Ports are those from 49152 through 65535. Well Known and Registered Ports are intended for use by server applications that desire a default contact point on a system. On most systems, Well Known Ports can only be used by system (or root) processes or by programs executed by privileged users, while Registered Ports can be used by ordinary user processes or programs executed by ordinary users. Dynamic and/or Private Ports are intended for temporary use, including client-side ports, out-of-band negotiated ports, and application testing prior to registration of a dedicated port; they MUST NOT be registered.

The Port Numbers registry should accept registrations for SCTP ports in the Well Known Ports and Registered Ports ranges. Well Known and Registered Ports SHOULD NOT be used without registration. Although in some cases -- such as porting an application from TCP to SCTP -- it may seem natural to use an SCTP port before registration completes, we emphasize that IANA will not guarantee registration of particular Well Known and Registered Ports. Registrations should be requested as early as possible.

Each port registration SHALL include the following information:

- o A short port name, consisting entirely of letters (A-Z and a-z), digits (0-9), and punctuation characters from "-_+./*" (not including the quotes).
- o The port number that is requested for registration.
- o A short English phrase describing the port's purpose.
- o Name and contact information for the person or entity performing the registration, and possibly a reference to a document defining the port's use. Registrations coming from IETF working groups need only name the working group, but indicating a contact person is recommended.

Registrants are encouraged to follow these guidelines when submitting a registration.

- o A port name SHOULD NOT be registered for more than one SCTP port


```

|  Type = 0      | Reserved|U|B|E|      Length      |
+-----+-----+-----+-----+-----+-----+
|                                     TSN                                     |
+-----+-----+-----+-----+-----+-----+
|  Stream Identifier S      | Stream Sequence Number n      |
+-----+-----+-----+-----+-----+-----+
|                                     Payload Protocol Identifier                                     |
+-----+-----+-----+-----+-----+-----+
\                                                                                   \
/                                     User Data (seq n of Stream S)                  /
\                                                                                   \
+-----+-----+-----+-----+-----+-----+

```

Reserved: 5 bits

Should be set to all '0's and ignored by the receiver.

New text: (Section 3.3.1)

```

      0           1           2           3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+
|  Type = 0      | Res  |I|U|B|E|      Length      |
+-----+-----+-----+-----+-----+-----+
|                                     TSN                                     |
+-----+-----+-----+-----+-----+-----+
|  Stream Identifier S      | Stream Sequence Number n      |
+-----+-----+-----+-----+-----+-----+
|                                     Payload Protocol Identifier                                     |
+-----+-----+-----+-----+-----+-----+
\                                                                                   \
/                                     User Data (seq n of Stream S)                  /
\                                                                                   \
+-----+-----+-----+-----+-----+-----+

```

Res: 4 bits

SHOULD be set to all '0's and ignored by the receiver.

I bit: 1 bit

The (I)mmmediate Bit MAY be set by the sender, whenever the sender of a DATA chunk can benefit from the corresponding SACK chunk being sent back without delay. See [RFC7053] for a discussion about

This text is in final form, and is not further updated in this document.

New text: (Append to Section 6.1)

Whenever the sender of a DATA chunk can benefit from the corresponding SACK chunk being sent back without delay, the sender MAY set the I bit in the DATA chunk header. Please note that why the sender has set the I bit is irrelevant to the receiver.

Reasons for setting the I bit include, but are not limited to (see Section 4 of [RFC7053] for the benefits):

- o The application requests to set the I bit of the last DATA chunk of a user message when providing the user message to the SCTP implementation (see Section 7).
- o The sender is in the SHUTDOWN-PENDING state.
- o The sending of a DATA chunk fills the congestion or receiver window.

This text is in final form, and is not further updated in this document.

Old text: (Section 6.2)

Note: The SHUTDOWN chunk does not contain Gap Ack Block fields. Therefore, the endpoint should use a SACK instead of the SHUTDOWN chunk to acknowledge DATA chunks received out of order.

New text: (Section 6.2)

Note: The SHUTDOWN chunk does not contain Gap Ack Block fields. Therefore, the endpoint SHOULD use a SACK instead of the SHUTDOWN chunk to acknowledge DATA chunks received out of order.

Upon receipt of an SCTP packet containing a DATA chunk with the I bit set, the receiver SHOULD NOT delay the sending of the corresponding SACK chunk, i.e., the receiver SHOULD immediately respond with the corresponding SACK chunk.

Please note that this change is only about adding a paragraph.

This text is in final form, and is not further updated in this document.

Old text: (Section 10.1 E)

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

New text: (Section 10.1 E)

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]
[,sack immediately])
-> result

This text is in final form, and is not further updated in this document.

New text: (Append optional parameter in Subsection E of Section 10.1)

- o sack immediately - set the I bit on the last DATA chunk used for sending buffer.

This text is in final form, and is not further updated in this document.

3.45.3. Solution Description

[RFC7053] was integrated.

3.46. CRC32c Code Improvements

3.46.1. Description of the Problem

The code given for the CRC32c computations uses types like long which may have different length on different operating systems or processors. Therefore, the code is changed to use specific types like uint32_t.

While there, fix also some syntax errors and a comment.

3.46.2. Text Changes to the Document

```

-----
Old text: (Appendix C)
-----
/*****
/* Note Definition for Ross Williams table generator would */
/* be: TB_WIDTH=4, TB_POLLY=0x1EDC6F41, TB_REVER=TRUE */
/* For Mr. Williams direct calculation code use the settings */
/* cm_width=32, cm_poly=0x1EDC6F41, cm_init=0xFFFFFFFF, */
/* cm_refin=TRUE, cm_refot=TRUE, cm_xorort=0x00000000 */
*****/

/* Example of the crc table file */
#ifndef __crc32cr_table_h__
#define __crc32cr_table_h__

#define CRC32C_POLY 0x1EDC6F41
#define CRC32C(c,d) (c=(c>>8)^crc_c[(c^(d))&0xFF])

unsigned long crc_c[256] =
{
0x00000000L, 0xF26B8303L, 0xE13B70F7L, 0x1350F3F4L,
0xC79A971FL, 0x35F1141CL, 0x26A1E7E8L, 0xD4CA64EBL,
0x8AD958CFL, 0x78B2DBCCL, 0x6BE22838L, 0x9989AB3BL,
0x4D43CFD0L, 0xBF284CD3L, 0xAC78BF27L, 0x5E133C24L,
0x105EC76FL, 0xE235446CL, 0xF165B798L, 0x030E349BL,
0xD7C45070L, 0x25AFD373L, 0x36FF2087L, 0xC494A384L,
0x9A879FA0L, 0x68EC1CA3L, 0x7BBCEF57L, 0x89D76C54L,
0x5D1D08BFL, 0xAF768BBCL, 0xBC267848L, 0x4E4DFB4BL,
0x20BD8EDEL, 0xD2D60DDDL, 0xC186FE29L, 0x33ED7D2AL,
0xE72719C1L, 0x154C9AC2L, 0x061C6936L, 0xF477EA35L,
0xAA64D611L, 0x580F5512L, 0x4B5FA6E6L, 0xB93425E5L,
0x6DFE410EL, 0x9F95C20DL, 0x8CC531F9L, 0x7EAE2FAL,
0x30E349B1L, 0xC288CAB2L, 0xD1D83946L, 0x23B3BA45L,

```

0xF779DEAEL, 0x05125DADL, 0x1642AE59L, 0xE4292D5AL,
0xBA3A117EL, 0x4851927DL, 0x5B016189L, 0xA96AE28AL,
0x7DA08661L, 0x8FCB0562L, 0x9C9BF696L, 0x6EF07595L,
0x417B1DBCL, 0xB3109EBFL, 0xA0406D4BL, 0x522BEE48L,
0x86E18AA3L, 0x748A09A0L, 0x67DAFA54L, 0x95B17957L,
0xCBA24573L, 0x39C9C670L, 0x2A993584L, 0xD8F2B687L,
0x0C38D26CL, 0xFE53516FL, 0xED03A29BL, 0x1F682198L,
0x5125DAD3L, 0xA34E59D0L, 0xB01EAA24L, 0x42752927L,
0x96BF4DCCL, 0x64D4CECFL, 0x77843D3BL, 0x85EFBE38L,
0xDBFC821CL, 0x2997011FL, 0x3AC7F2EBL, 0xC8AC71E8L,
0x1C661503L, 0xEE0D9600L, 0xFD5D65F4L, 0x0F36E6F7L,
0x61C69362L, 0x93AD1061L, 0x80FDE395L, 0x72966096L,
0xA65C047DL, 0x5437877EL, 0x4767748AL, 0xB50CF789L,
0xEB1FCBADL, 0x197448AEL, 0x0A24BB5AL, 0xF84F3859L,
0x2C855CB2L, 0xDEEEDFB1L, 0xCDBE2C45L, 0x3FD5AF46L,
0x7198540DL, 0x83F3D70EL, 0x90A324FAL, 0x62C8A7F9L,
0xB602C312L, 0x44694011L, 0x5739B3E5L, 0xA55230E6L,
0xFB410CC2L, 0x092A8FC1L, 0x1A7A7C35L, 0xE811FF36L,
0x3CDB9BDDL, 0xCEB018DEL, 0xDDE0EB2AL, 0x2F8B6829L,
0x82F63B78L, 0x709DB87BL, 0x63CD4B8FL, 0x91A6C88CL,
0x456CAC67L, 0xB7072F64L, 0xA457DC90L, 0x563C5F93L,
0x082F63B7L, 0xFA44E0B4L, 0xE9141340L, 0x1B7F9043L,
0xCFB5F4A8L, 0x3DDE77ABL, 0x2E8E845FL, 0xDCE5075CL,
0x92A8FC17L, 0x60C37F14L, 0x73938CE0L, 0x81F80FE3L,
0x55326B08L, 0xA759E80BL, 0xB4091BFFL, 0x466298FCL,
0x1871A4D8L, 0xEA1A27DBL, 0xF94AD42FL, 0x0B21572CL,
0xDFEB33C7L, 0x2D80B0C4L, 0x3ED04330L, 0xCCBBC033L,
0xA24BB5A6L, 0x502036A5L, 0x4370C551L, 0xB11B4652L,
0x65D122B9L, 0x97BAA1BAL, 0x84EA524EL, 0x7681D14DL,
0x2892ED69L, 0xDAF96E6AL, 0xC9A99D9EL, 0x3BC21E9DL,
0xEF087A76L, 0x1D63F975L, 0x0E330A81L, 0xFC588982L,
0xB21572C9L, 0x407EF1CAL, 0x532E023EL, 0xA145813DL,
0x758FE5D6L, 0x87E466D5L, 0x94B49521L, 0x66DF1622L,
0x38CC2A06L, 0xCAA7A905L, 0xD9F75AF1L, 0x2B9CD9F2L,
0xFF56BD19L, 0x0D3D3E1AL, 0x1E6DCDEEL, 0xEC064EEDL,
0xC38D26C4L, 0x31E6A5C7L, 0x22B65633L, 0xD0DDD530L,
0x0417B1DBL, 0xF67C32D8L, 0xE52CC12CL, 0x1747422FL,
0x49547E0BL, 0xBB3FFD08L, 0xA86F0EFCL, 0x5A048DFFL,
0x8ECEE914L, 0x7CA56A17L, 0x6FF599E3L, 0x9D9E1AE0L,
0xD3D3E1ABL, 0x21B862A8L, 0x32E8915CL, 0xC083125FL,
0x144976B4L, 0xE622F5B7L, 0xF5720643L, 0x07198540L,
0x590AB964L, 0xAB613A67L, 0xB831C993L, 0x4A5A4A90L,
0x9E902E7BL, 0x6CFBAD78L, 0x7FAB5E8CL, 0x8DC0DD8FL,
0xE330A81AL, 0x115B2B19L, 0x020BD8EDL, 0xF0605BEEL,
0x24AA3F05L, 0xD6C1BC06L, 0xC5914FF2L, 0x37FACCF1L,
0x69E9F0D5L, 0x9B8273D6L, 0x88D28022L, 0x7AB90321L,
0xAE7367CAL, 0x5C18E4C9L, 0x4F48173DL, 0xBD23943EL,
0xF36E6F75L, 0x0105EC76L, 0x12551F82L, 0xE03E9C81L,

```

0x34F4F86AL, 0xC69F7B69L, 0xD5CF889DL, 0x27A40B9EL,
0x79B737BAL, 0x8BDCB4B9L, 0x988C474DL, 0x6AE7C44EL,
0xBE2DA0A5L, 0x4C4623A6L, 0x5F16D052L, 0xAD7D5351L,
};

#endif

```

```

-----
New text: (Appendix B)
-----

```

```

<CODE BEGINS>
/*****
/* Note Definition for Ross Williams table generator would */
/* be: TB_WIDTH=4, TB_POLLY=0x1EDC6F41, TB_REVER=TRUE */
/* For Mr. Williams direct calculation code use the settings */
/* cm_width=32, cm_poly=0x1EDC6F41, cm_init=0xFFFFFFFF, */
/* cm_refin=TRUE, cm_refot=TRUE, cm_xorort=0x00000000 */
*****/

/* Example of the crc table file */
#ifndef __crc32cr_h__
#define __crc32cr_h__

#define CRC32C_POLY 0x1EDC6F41UL
#define CRC32C(c,d) (c=(c>>8)^crc_c[(c^(d))&0xFF])

uint32_t crc_c[256] =
{
0x00000000UL, 0xF26B8303UL, 0xE13B70F7UL, 0x1350F3F4UL,
0xC79A971FUL, 0x35F1141CUL, 0x26A1E7E8UL, 0xD4CA64EBUL,
0x8AD958CFUL, 0x78B2DBCCUL, 0x6BE22838UL, 0x9989AB3BUL,
0x4D43CFD0UL, 0xBF284CD3UL, 0xAC78BF27UL, 0x5E133C24UL,
0x105EC76FUL, 0xE235446CUL, 0xF165B798UL, 0x030E349BUL,
0xD7C45070UL, 0x25AFD373UL, 0x36FF2087UL, 0xC494A384UL,
0x9A879FA0UL, 0x68EC1CA3UL, 0x7BBCEF57UL, 0x89D76C54UL,
0x5D1D08BFUL, 0xAF768BBCUL, 0xBC267848UL, 0x4E4DFB4BUL,
0x20BD8EDEUL, 0xD2D60DDDUL, 0xC186FE29UL, 0x33ED7D2AUL,
0xE72719C1UL, 0x154C9AC2UL, 0x061C6936UL, 0xF477EA35UL,
0xAA64D611UL, 0x580F5512UL, 0x4B5FA6E6UL, 0xB93425E5UL,
0x6DFE410EUL, 0x9F95C20DUL, 0x8CC531F9UL, 0x7EAE82FAUL,
0x30E349B1UL, 0xC288CAB2UL, 0xD1D83946UL, 0x23B3BA45UL,
0xF779DEAEUL, 0x05125DADUL, 0x1642AE59UL, 0xE4292D5AUL,
0xBA3A117EUL, 0x4851927DUL, 0x5B016189UL, 0xA96AE28AUL,
0x7DA08661UL, 0x8FCB0562UL, 0x9C9BF696UL, 0x6EF07595UL,
0x417B1DBCUL, 0xB3109EBFUL, 0xA0406D4BUL, 0x522BEE48UL,
0x86E18AA3UL, 0x748A09A0UL, 0x67DAFA54UL, 0x95B17957UL,
0xCBA24573UL, 0x39C9C670UL, 0x2A993584UL, 0xD8F2B687UL,

```

```
0x0C38D26CUL, 0xFE53516FUL, 0xED03A29BUL, 0x1F682198UL,  
0x5125DAD3UL, 0xA34E59D0UL, 0xB01EAA24UL, 0x42752927UL,  
0x96BF4DCCUL, 0x64D4CECFUL, 0x77843D3BUL, 0x85EFBE38UL,  
0xDBFC821CUL, 0x2997011FUL, 0x3AC7F2EBUL, 0xC8AC71E8UL,  
0x1C661503UL, 0xEE0D9600UL, 0xFD5D65F4UL, 0x0F36E6F7UL,  
0x61C69362UL, 0x93AD1061UL, 0x80FDE395UL, 0x72966096UL,  
0xA65C047DUL, 0x5437877EUL, 0x4767748AUL, 0xB50CF789UL,  
0xEB1FCBADUL, 0x197448AEUL, 0x0A24BB5AUL, 0xF84F3859UL,  
0x2C855CB2UL, 0xDEEEDFB1UL, 0xCDBE2C45UL, 0x3FD5AF46UL,  
0x7198540DUL, 0x83F3D70EUL, 0x90A324FAUL, 0x62C8A7F9UL,  
0xB602C312UL, 0x44694011UL, 0x5739B3E5UL, 0xA55230E6UL,  
0xFB410CC2UL, 0x092A8FC1UL, 0x1A7A7C35UL, 0xE811FF36UL,  
0x3CDB9BDDUL, 0xCEB018DEUL, 0xDDE0EB2AUL, 0x2F8B6829UL,  
0x82F63B78UL, 0x709DB87BUL, 0x63CD4B8FUL, 0x91A6C88CUL,  
0x456CAC67UL, 0xB7072F64UL, 0xA457DC90UL, 0x563C5F93UL,  
0x082F63B7UL, 0xFA44E0B4UL, 0xE9141340UL, 0x1B7F9043UL,  
0xCFB5F4A8UL, 0x3DDE77ABUL, 0x2E8E845FUL, 0xDCE5075CUL,  
0x92A8FC17UL, 0x60C37F14UL, 0x73938CE0UL, 0x81F80FE3UL,  
0x55326B08UL, 0xA759E80BUL, 0xB4091BFFUL, 0x466298FCUL,  
0x1871A4D8UL, 0xEA1A27DBUL, 0xF94AD42FUL, 0x0B21572CUL,  
0xDFEB33C7UL, 0x2D80B0C4UL, 0x3ED04330UL, 0xCCBBC033UL,  
0xA24BB5A6UL, 0x502036A5UL, 0x4370C551UL, 0xB11B4652UL,  
0x65D122B9UL, 0x97BAA1BAUL, 0x84EA524EUL, 0x7681D14DUL,  
0x2892ED69UL, 0xDAF96E6AUL, 0xC9A99D9EUL, 0x3BC21E9DUL,  
0xEF087A76UL, 0x1D63F975UL, 0x0E330A81UL, 0xFC588982UL,  
0xB21572C9UL, 0x407EF1CAUL, 0x532E023EUL, 0xA145813DUL,  
0x758FE5D6UL, 0x87E466D5UL, 0x94B49521UL, 0x66DF1622UL,  
0x38CC2A06UL, 0xCA7A905UL, 0xD9F75AF1UL, 0x2B9CD9F2UL,  
0xFF56BD19UL, 0x0D3D3E1AUL, 0x1E6DCDEEUL, 0xEC064EEDUL,  
0xC38D26C4UL, 0x31E6A5C7UL, 0x22B65633UL, 0xD0DDD530UL,  
0x0417B1DBUL, 0xF67C32D8UL, 0xE52CC12CUL, 0x1747422FUL,  
0x49547E0BUL, 0xBB3FFD08UL, 0xA86F0EFCUL, 0x5A048DFFUL,  
0x8ECEE914UL, 0x7CA56A17UL, 0x6FF599E3UL, 0x9D9E1AE0UL,  
0xD3D3E1ABUL, 0x21B862A8UL, 0x32E8915CUL, 0xC083125FUL,  
0x144976B4UL, 0xE622F5B7UL, 0xF5720643UL, 0x07198540UL,  
0x590AB964UL, 0xAB613A67UL, 0xB831C993UL, 0x4A5A4A90UL,  
0x9E902E7BUL, 0x6CFBAD78UL, 0x7FAB5E8CUL, 0x8DC0DD8FUL,  
0xE330A81AUL, 0x115B2B19UL, 0x020BD8EDUL, 0xF0605BEEUL,  
0x24AA3F05UL, 0xD6C1BC06UL, 0xC5914FF2UL, 0x37FACCF1UL,  
0x69E9F0D5UL, 0x9B8273D6UL, 0x88D28022UL, 0x7AB90321UL,  
0xAE7367CAUL, 0x5C18E4C9UL, 0x4F48173DUL, 0xBD23943EUL,  
0xF36E6F75UL, 0x0105EC76UL, 0x12551F82UL, 0xE03E9C81UL,  
0x34F4F86AUL, 0xC69F7B69UL, 0xD5CF889DUL, 0x27A40B9EUL,  
0x79B737BAUL, 0x8BDCB4B9UL, 0x988C474DUL, 0x6AE7C44EUL,  
0xBE2DA0A5UL, 0x4C4623A6UL, 0x5F16D052UL, 0xAD7D5351UL,  
};
```

```
#endif
```


This text has been modified by multiple errata. It includes modifications from Section 3.10. It is in final form, and is not further updated in this document.

Old text: (Appendix C)

```
/* Example of table build routine */

#include <stdio.h>
#include <stdlib.h>

#define OUTPUT_FILE    "crc32cr.h"
#define CRC32C_POLY    0x1EDC6F41L
FILE *tf;
unsigned long
reflect_32 (unsigned long b)
{
    int i;
    unsigned long rw = 0L;

    for (i = 0; i < 32; i++){
        if (b & 1)
            rw |= 1 << (31 - i);
        b >>= 1;
    }
    return (rw);
}

unsigned long
build_crc_table (int index)
{
    int i;
    unsigned long rb;

    rb = reflect_32 (index);

    for (i = 0; i < 8; i++){
        if (rb & 0x80000000L)
            rb = (rb << 1) ^ CRC32C_POLY;
        else
            rb <<= 1;
    }
    return (reflect_32 (rb));
}
```

```

main ()
{
    int i;

    printf ("\nGenerating CRC-32c table file <%=s>\n",
        OUTPUT_FILE);
    if ((tf = fopen (OUTPUT_FILE, "w")) == NULL){
        printf ("Unable to open %s\n", OUTPUT_FILE);
        exit (1);
    }
    fprintf (tf, "#ifndef __crc32cr_table_h__\n");
    fprintf (tf, "#define __crc32cr_table_h__\n\n");
    fprintf (tf, "#define CRC32C_POLY 0x%08lX\n",
        CRC32C_POLY);
    fprintf (tf,
        "#define CRC32C(c,d) (c=(c>>8)^crc_c[(c^(d))&0xFF])\n");
    fprintf (tf, "\nunsigned long  crc_c[256] =\n{\n");
    for (i = 0; i < 256; i++){
        fprintf (tf, "0x%08lXL, ", build_crc_table (i));
        if ((i & 3) == 3)
            fprintf (tf, "\n");
    }
    fprintf (tf, "};\n\n#endif\n");

    if (fclose (tf) != 0)
        printf ("Unable to close <%=s>." OUTPUT_FILE);
    else
        printf ("\nThe CRC-32c table has been written to <%=s>.\n",
            OUTPUT_FILE);
}

```

```

-----
New text: (Appendix B)
-----

```

```

/* Example of table build routine */

#include <stdio.h>
#include <stdlib.h>

#define OUTPUT_FILE    "crc32cr.h"
#define CRC32C_POLY    0x1EDC6F41UL

static FILE *tf;

static uint32_t
reflect_32(uint32_t b)
{

```

```

    int i;
    uint32_t rw = 0UL;

    for (i = 0; i < 32; i++) {
        if (b & 1)
            rw |= 1 << (31 - i);
        b >>= 1;
    }
    return (rw);
}

static uint32_t
build_crc_table(int index)
{
    int i;
    uint32_t rb;

    rb = reflect_32(index);

    for (i = 0; i < 8; i++) {
        if (rb & 0x80000000UL)
            rb = (rb << 1) ^ (uint32_t)CRC32C_POLY;
        else
            rb <<= 1;
    }
    return (reflect_32(rb));
}

int
main (void)
{
    int i;

    printf("\nGenerating CRC-32c table file <%=s>\n",
        OUTPUT_FILE);
    if ((tf = fopen(OUTPUT_FILE, "w")) == NULL) {
        printf ("Unable to open %s\n", OUTPUT_FILE);
        exit (1);
    }
    fprintf(tf, "#ifndef __crc32cr_h__\n");
    fprintf(tf, "#define __crc32cr_h__\n\n");
    fprintf(tf, "#define CRC32C_POLY 0x%08XUL\n",
        (uint32_t)CRC32C_POLY);
    fprintf(tf,
        "#define CRC32C(c,d) (c=(c>>8)^crc_c[(c^(d))&0xFF])\n");
    fprintf(tf, "\nuint32_t crc_c[256] =\n{\n");
    for (i = 0; i < 256; i++) {
        fprintf(tf, "0x%08XUL,", build_crc_table (i));

```

```
        if ((i & 3) == 3)
            fprintf(tf, "\n");
        else
            fprintf(tf, " ");
    }
    fprintf(tf, "};\n\n#endif\n");

    if (fclose (tf) != 0)
        printf("Unable to close <%s>.", OUTPUT_FILE);
    else
        printf("\nThe CRC-32c table has been written to <%s>.\n",
            OUTPUT_FILE);
}
```

This text has been modified by multiple errata. It includes modifications from Section 3.10. It is in final form, and is not further updated in this document.

Old text: (Appendix C)

```
/* Example of crc insertion */

#include "crc32cr.h"

unsigned long
generate_crc32c(unsigned char *buffer, unsigned int length)
{
    unsigned int i;
    unsigned long crc32 = ~0L;
    unsigned long result;
    unsigned char byte0,byte1,byte2,byte3;

    for (i = 0; i < length; i++){
        CRC32C(crc32, buffer[i]);
    }

    result = ~crc32;

    /* result now holds the negated polynomial remainder;
     * since the table and algorithm is "reflected" [williams95].
     * That is, result has the same value as if we mapped the message
     * to a polynomial, computed the host-bit-order polynomial
     * remainder, performed final negation, then did an end-for-end
     * bit-reversal.
     */
}
```

```
* Note that a 32-bit bit-reversal is identical to four inplace
* 8-bit reversals followed by an end-for-end byteswap.
* In other words, the bytes of each bit are in the right order,
* but the bytes have been byteswapped. So we now do an explicit
* byteswap. On a little-endian machine, this byteswap and
* the final ntohl cancel out and could be elided.
*/

byte0 = result & 0xff;
byte1 = (result>>8) & 0xff;
byte2 = (result>>16) & 0xff;
byte3 = (result>>24) & 0xff;
crc32 = ((byte0 << 24) |
         (byte1 << 16) |
         (byte2 << 8)  |
         byte3);
return ( crc32 );
}

int
insert_crc32(unsigned char *buffer, unsigned int length)
{
    SCTP_message *message;
    unsigned long crc32;
    message = (SCTP_message *) buffer;
    message->common_header.checksum = 0L;
    crc32 = generate_crc32c(buffer,length);
    /* and insert it into the message */
    message->common_header.checksum = htonl(crc32);
    return 1;
}

int
validate_crc32(unsigned char *buffer, unsigned int length)
{
    SCTP_message *message;
    unsigned int i;
    unsigned long original_crc32;
    unsigned long crc32 = ~0L;

    /* save and zero checksum */
    message = (SCTP_message *) buffer;
    original_crc32 = ntohl(message->common_header.checksum);
    message->common_header.checksum = 0L;
    crc32 = generate_crc32c(buffer,length);
    return ((original_crc32 == crc32)? 1 : -1);
}
```

```
-----
New text: (Appendix B)
-----

/* Example of crc insertion */

#include "crc32cr.h"

uint32_t
generate_crc32c(unsigned char *buffer, unsigned int length)
{
    unsigned int i;
    uint32_t crc32 = 0xffffffffUL;
    uint32_t result;
    uint8_t byte0, byte1, byte2, byte3;

    for (i = 0; i < length; i++) {
        CRC32C(crc32, buffer[i]);
    }

    result = ~crc32;

    /* result now holds the negated polynomial remainder;
     * since the table and algorithm is "reflected" [williams95].
     * That is, result has the same value as if we mapped the message
     * to a polynomial, computed the host-bit-order polynomial
     * remainder, performed final negation, then did an end-for-end
     * bit-reversal.
     * Note that a 32-bit bit-reversal is identical to four inplace
     * 8-bit reversals followed by an end-for-end byteswap.
     * In other words, the bits of each byte are in the right order,
     * but the bytes have been byteswapped. So we now do an explicit
     * byteswap. On a little-endian machine, this byteswap and
     * the final ntohl cancel out and could be elided.
     */

    byte0 = result & 0xff;
    byte1 = (result>>8) & 0xff;
    byte2 = (result>>16) & 0xff;
    byte3 = (result>>24) & 0xff;
    crc32 = ((byte0 << 24) |
              (byte1 << 16) |
              (byte2 << 8) |
              byte3);
    return (crc32);
}

int
```

```
insert_crc32(unsigned char *buffer, unsigned int length)
{
    SCTP_message *message;
    uint32_t crc32;
    message = (SCTP_message *) buffer;
    message->common_header.checksum = 0UL;
    crc32 = generate_crc32c(buffer, length);
    /* and insert it into the message */
    message->common_header.checksum = htonl(crc32);
    return 1;
}

int
validate_crc32(unsigned char *buffer, unsigned int length)
{
    SCTP_message *message;
    unsigned int i;
    uint32_t original_crc32;
    uint32_t crc32;

    /* save and zero checksum */
    message = (SCTP_message *)buffer;
    original_crc32 = ntohl(message->common_header.checksum);
    message->common_header.checksum = 0L;
    crc32 = generate_crc32c(buffer, length);
    return ((original_crc32 == crc32)? 1 : -1);
}
<CODE ENDS>
```

This text has been modified by multiple errata. It includes modifications from Section 3.5 and Section 3.10. It is in final form, and is not further updated in this document.

3.46.3. Solution Description

The code was changed to use platform independent types.

3.47. Clarification of Gap Ack Blocks in SACK Chunks

3.47.1. Description of the Problem

The Gap Ack Blocks in the SACK chunk are intended to be isolated. However, this is not mentioned with normative text.

This issue was reported as part of an Errata for [RFC4960] with Errata ID 5202.

3.47.2. Text Changes to the Document

Old text: (Section 3.3.4)

The SACK also contains zero or more Gap Ack Blocks. Each Gap Ack Block acknowledges a subsequence of TSNs received following a break in the sequence of received TSNs. By definition, all TSNs acknowledged by Gap Ack Blocks are greater than the value of the Cumulative TSN Ack.

New text: (Section 3.3.4)

The SACK also contains zero or more Gap Ack Blocks. Each Gap Ack Block acknowledges a subsequence of TSNs received following a break in the sequence of received TSNs. The Gap Ack Blocks SHOULD be isolated. This means that the TSN just before each Gap Ack Block and the TSN just after each Gap Ack Block has not been received. By definition, all TSNs acknowledged by Gap Ack Blocks are greater than the value of the Cumulative TSN Ack.

This text is in final form, and is not further updated in this document.

Old text: (Section 3.3.4)

Gap Ack Blocks:

These fields contain the Gap Ack Blocks. They are repeated for each Gap Ack Block up to the number of Gap Ack Blocks defined in the Number of Gap Ack Blocks field. All DATA chunks with TSNs greater than or equal to (Cumulative TSN Ack + Gap Ack Block Start) and less than or equal to (Cumulative TSN Ack + Gap Ack Block End) of each Gap Ack Block are assumed to have been received correctly.

New text: (Section 3.3.4)

Gap Ack Blocks:

These fields contain the Gap Ack Blocks. They are repeated for each Gap Ack Block up to the number of Gap Ack Blocks defined in the Number of Gap Ack Blocks field. All DATA chunks with TSNs greater than or equal to (Cumulative TSN Ack + Gap Ack Block Start) and less than or equal to (Cumulative TSN Ack + Gap Ack Block End) of each Gap Ack Block are assumed to have been received correctly. Gap Ack Blocks SHOULD be isolated. That means that the DATA chunks with TSN equal to (Cumulative TSN Ack + Gap Ack Block Start - 1) and (Cumulative TSN Ack + Gap Ack Block End + 1) have not been received.

This text is in final form, and is not further updated in this document.

3.47.3. Solution Description

Normative text describing the intended usage of Gap Ack Blocks has been added.

3.48. Handling of SSN Wrap Arounds

3.48.1. Description of the Problem

The Stream Sequence Number (SSN) is used for preserving the ordering of user messages within each SCTP stream. The SSN is limited to 16 bits. Therefore, multiple wrap arounds of the SSN might happen within the current send window. To allow the receiver to deliver

ordered user messages in the correct sequence, the sender should limit the number of user messages per stream.

3.48.2. Text Changes to the Document

Old text: (Section 6.1)

Note: The data sender SHOULD NOT use a TSN that is more than $2^{31} - 1$ above the beginning TSN of the current send window.

New text: (Section 6.1)

Note: The data sender SHOULD NOT use a TSN that is more than $2^{31} - 1$ above the beginning TSN of the current send window.

Note: For each stream, the data sender SHOULD NOT have more than $2^{16} - 1$ ordered user messages in the current send window.

This text is in final form, and is not further updated in this document.

3.48.3. Solution Description

The data sender is required to limit the number of ordered user messages within the current send window.

3.49. Update RFC 2119 Boilerplate

3.49.1. Description of the Problem

The text to be used to refer to the [RFC2119] terms has been updated by [RFC8174].

3.49.2. Text Changes to the Document

Old text: (Section 2)

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].

New text: (Section 2)

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.

This text is in final form, and is not further updated in this document.

3.49.3. Solution Description

The text has been updated to the one specified in [RFC8174].

3.50. Missed Text Removal

3.50.1. Description of the Problem

When integrating the changes to Section 7.2.4 of [RFC2960] as described in Section 2.8.2 of [RFC4460] some text was not removed and is therefore still in [RFC4960].

3.50.2. Text Changes to the Document

Old text: (Section 7.2.4)

A straightforward implementation of the above keeps a counter for each TSN hole reported by a SACK. The counter increments for each consecutive SACK reporting the TSN hole. After reaching 3 and starting the Fast-Retransmit procedure, the counter resets to 0. Because cwnd in SCTP indirectly bounds the number of outstanding TSN's, the effect of TCP Fast Recovery is achieved automatically with no adjustment to the congestion control window size.

New text: (Section 7.2.4)

This text is in final form, and is not further updated in this document.

3.50.3. Solution Description

The text has finally been removed.

4. IANA Considerations

Section 3.44 of this document updates the port number registry for SCTP to be consistent with [RFC6335]. IANA is requested to review Section 3.44.

IANA is only requested to check if it is OK to make the proposed text change in an upcoming standards track document that updates [RFC4960]. IANA is not asked to perform any other action and this document does not request IANA to make a change to any registry.

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, Spencer Dawkins, Gorrry Fairhurst, Benjamin Kaduk, Mirja Kuehlewind, Peter Lei, Gyula Marosi, Lionel Morand, Jeff Morriss, Karen E. E. Nielsen, Tom Petch, Kacheong Poon, Julien Pourtet, Irene Ruengeler, Michael Welzl, and Qiaobing Xie 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, <<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>>.

7.2. Informative References

- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<https://www.rfc-editor.org/info/rfc1122>>.
- [RFC1858] Ziemba, G., Reed, D., and P. Traina, "Security Considerations for IP Fragment Filtering", RFC 1858, DOI 10.17487/RFC1858, October 1995, <<https://www.rfc-editor.org/info/rfc1858>>.
- [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, <<https://www.rfc-editor.org/info/rfc2960>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [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, <<https://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, <<https://www.rfc-editor.org/info/rfc5681>>.

- [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>>.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.
- [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, <<https://www.rfc-editor.org/info/rfc6335>>.
- [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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.
- [RFC8311] Black, D., "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation", RFC 8311, DOI 10.17487/RFC8311, January 2018, <<https://www.rfc-editor.org/info/rfc8311>>.

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

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/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 17, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. UDP Usage Guidelines	5
3.1. Congestion Control Guidelines	6
3.2. Message Size Guidelines	18
3.3. Reliability Guidelines	20
3.4. Checksum Guidelines	21
3.5. Middlebox Traversal Guidelines	24
3.6. Limited Applicability and Controlled Environments	26
4. Multicast UDP Usage Guidelines	27
4.1. Multicast Congestion Control Guidelines	29
4.2. Message Size Guidelines for Multicast	31
5. Programming Guidelines	31
5.1. Using UDP Ports	33
5.2. ICMP Guidelines	36
6. Security Considerations	36
7. Summary	38
8. IANA Considerations	40
9. Acknowledgments	41
10. References	41
10.1. Normative References	41
10.2. Informative References	43
Appendix A. Case Study of the Use of IPv6 UDP Zero-Checksum Mode	52
Appendix B. Revision Notes	53

Authors' Addresses	58
------------------------------	----

1. Introduction

The User Datagram Protocol (UDP) [RFC0768] provides a minimal, unreliable, best-effort, message-passing transport to applications and other protocols (such as tunnels) that desire to operate over IP. Both are simply called "applications" in the remainder of this document.

Compared to other transport protocols, UDP and its UDP-Lite variant [RFC3828] are unique in that they do not establish end-to-end connections between communicating end systems. UDP communication consequently does not incur connection establishment and teardown overheads, and there is minimal associated end system state. Because of these characteristics, UDP can offer a very efficient communication transport to some applications.

A second unique characteristic of UDP is that it provides no inherent congestion control mechanisms. On many platforms, applications can send UDP datagrams at the line rate of the platform's link interface, which is often much greater than the available end-to-end path capacity, and doing so contributes to congestion along the path. [RFC2914] describes the best current practice for congestion control in the Internet. It identifies two major reasons why congestion control mechanisms are critical for the stable operation of the Internet:

1. The prevention of congestion collapse, i.e., a state where an increase in network load results in a decrease in useful work done by the network.
2. The establishment of a degree of fairness, i.e., allowing multiple flows to share the capacity of a path reasonably equitably.

Because UDP itself provides no congestion control mechanisms, it is up to the applications that use UDP for Internet communication to employ suitable mechanisms to prevent congestion collapse and establish a degree of fairness. [RFC2309] discusses the dangers of congestion-unresponsive flows and states that "all UDP-based streaming applications should incorporate effective congestion avoidance mechanisms." [RFC7567] reaffirms this statement. This is an important requirement, even for applications that do not use UDP for streaming. In addition, congestion-controlled transmission is of benefit to an application itself, because it can reduce self-induced packet loss, minimize retransmissions, and hence reduce delays. Congestion control is essential even at relatively slow transmission

rates. For example, an application that generates five 1500-byte UDP datagrams in one second can already exceed the capacity of a 56 Kb/s path. For applications that can operate at higher, potentially unbounded data rates, congestion control becomes vital to prevent congestion collapse and establish some degree of fairness. Section 3 describes a number of simple guidelines for the designers of such applications.

A UDP datagram is carried in a single IP packet and is hence limited to a maximum payload of 65,507 bytes for IPv4 and 65,527 bytes for IPv6. The transmission of large IP packets usually requires IP fragmentation. Fragmentation decreases communication reliability and efficiency and should be avoided. IPv6 allows the option of transmitting large packets ("jumbograms") without fragmentation when all link layers along the path support this [RFC2675]. Some of the guidelines in Section 3 describe how applications should determine appropriate message sizes. Other sections of this document provide guidance on reliability, checksums, middlebox traversal and use of multicast.

This document provides guidelines and recommendations. Although most UDP applications are expected to follow these guidelines, there do exist valid reasons why a specific application may decide not to follow a given guideline. In such cases, it is RECOMMENDED that application designers cite the respective section(s) of this document in the technical specification of their application or protocol and explain their rationale for their design choice.

[RFC5405] was scoped to provide guidelines for unicast applications only, whereas this document also provides guidelines for UDP flows that use IP anycast, multicast, broadcast, and applications that use UDP tunnels to support IP flows.

Finally, although this document specifically refers to usage of UDP, the spirit of some of its guidelines also applies to other message-passing applications and protocols (specifically on the topics of congestion control, message sizes, and reliability). Examples include signaling, tunnel, or control applications that choose to run directly over IP by registering their own IP protocol number with IANA. This document is expected to provide useful background reading to the designers of such applications and protocols.

2. Terminology

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

3. UDP Usage Guidelines

Internet paths can have widely varying characteristics, including transmission delays, available bandwidths, congestion levels, reordering probabilities, supported message sizes, or loss rates. Furthermore, the same Internet path can have very different conditions over time. Consequently, applications that may be used on the Internet **MUST NOT** make assumptions about specific path characteristics. They **MUST** instead use mechanisms that let them operate safely under very different path conditions. Typically, this requires conservatively probing the current conditions of the Internet path they communicate over to establish a transmission behavior that it can sustain and that is reasonably fair to other traffic sharing the path.

These mechanisms are difficult to implement correctly. For most applications, the use of one of the existing IETF transport protocols is the simplest method of acquiring the required mechanisms. Doing so also avoids issues that protocols using a new IP protocol number face when being deployed over the Internet, where middleboxes that only support TCP and UDP are sometimes present. Consequently, the RECOMMENDED alternative to the UDP usage described in the remainder of this section is the use of an IETF transport protocol such as TCP [RFC0793], Stream Control Transmission Protocol (SCTP) [RFC4960], and SCTP Partial Reliability Extension (SCTP-PR) [RFC3758], or Datagram Congestion Control Protocol (DCCP) [RFC4340] with its different congestion control types [RFC4341][RFC4342][RFC5622], or transport protocols specified by the IETF in the future. (UDP-encapsulated SCTP [RFC6951] and DCCP [RFC6773] can offer support for traversing firewalls and other middleboxes where the native protocols are not supported.)

If used correctly, these more fully-featured transport protocols are not as "heavyweight" as often claimed. For example, the TCP algorithms have been continuously improved over decades, and have reached a level of efficiency and correctness that custom application-layer mechanisms will struggle to easily duplicate. In addition, many TCP implementations allow connections to be tuned by an application to its purposes. For example, TCP's "Nagle" algorithm [RFC1122] can be disabled, improving communication latency at the expense of more frequent -- but still congestion-controlled -- packet transmissions. Another example is the TCP SYN cookie mechanism [RFC4987], which is available on many platforms. TCP with SYN cookies does not require a server to maintain per-connection state until the connection is established. TCP also requires the end that closes a connection to maintain the TIME-WAIT state that prevents delayed segments from one connection instance from interfering with a later one. Applications that are aware of and designed for this

behavior can shift maintenance of the TIME-WAIT state to conserve resources by controlling which end closes a TCP connection [FABER]. Finally, TCP's built-in capacity-probing and awareness of the maximum transmission unit supported by the path (PMTU) results in efficient data transmission that quickly compensates for the initial connection setup delay, in the case of transfers that exchange more than a few segments.

3.1. Congestion Control Guidelines

If an application or protocol chooses not to use a congestion-controlled transport protocol, it SHOULD control the rate at which it sends UDP datagrams to a destination host, in order to fulfill the requirements of [RFC2914]. It is important to stress that an application SHOULD perform congestion control over all UDP traffic it sends to a destination, independently from how it generates this traffic. For example, an application that forks multiple worker processes or otherwise uses multiple sockets to generate UDP datagrams SHOULD perform congestion control over the aggregate traffic.

Several approaches to perform congestion control are discussed in the remainder of this section. The section describes generic topics with an intended emphasis on unicast and anycast [RFC1546] usage. Not all approaches discussed below are appropriate for all UDP-transmitting applications. Section 3.1.2 discusses congestion control options for applications that perform bulk transfers over UDP. Such applications can employ schemes that sample the path over several subsequent round-trips during which data is exchanged to determine a sending rate that the path at its current load can support. Other applications only exchange a few UDP datagrams with a destination. Section 3.1.3 discusses congestion control options for such "low data-volume" applications. Because they typically do not transmit enough data to iteratively sample the path to determine a safe sending rate, they need to employ different kinds of congestion control mechanisms. Section 3.1.11 discusses congestion control considerations when UDP is used as a tunneling protocol. Section 4 provides additional recommendations for broadcast and multicast usage.

It is important to note that congestion control should not be viewed as an add-on to a finished application. Many of the mechanisms discussed in the guidelines below require application support to operate correctly. Application designers need to consider congestion control throughout the design of their application, similar to how they consider security aspects throughout the design process.

In the past, the IETF has also investigated integrated congestion control mechanisms that act on the traffic aggregate between two hosts, i.e., a framework such as the Congestion Manager [RFC3124], where active sessions may share current congestion information in a way that is independent of the transport protocol. Such mechanisms have currently failed to see deployment, but would otherwise simplify the design of congestion control mechanisms for UDP sessions, so that they fulfill the requirements in [RFC2914].

3.1.1. Protocol Timer Guidelines

Understanding the latency between communicating endpoints is usually a crucial part of effective congestion control implementations for protocols and applications. Latency estimation can be used in a number of protocol functions, such as calculating a congestion-controlled transmission rate, triggering retransmission, and detecting packet loss. Additional protocol functions, for example, determining an interval for probing a path, determining an interval between keep-alive messages, determining an interval for measuring the quality of experience, or determining if a remote endpoint has responded to a request to perform an action typically operate over longer timescales than congestion control and therefore are not covered in this section.

The general recommendation in this document is that applications SHOULD leverage existing congestion control techniques and the latency estimators specified therein (see next subsection). The following guidelines are provided for applications that need to design their own latency estimation mechanisms.

The guidelines are framed in terms of "latency" and not "round-trip time" because some situations require characterizing only the network-based latency (e.g., TCP-Friendly Rate Control [RFC5348]), while other cases necessitate inclusion of the time required by the remote endpoint to provide feedback (e.g., developing an understanding of when to retransmit a message).

The latency between endpoints is generally a dynamic property. Therefore, estimates SHOULD represent some sort of averaging of multiple recent measurement samples to account for variance. Leveraging an Exponentially Weighted Moving Average (EWMA) has proven useful for this purpose (e.g., in TCP [RFC6298] and TCP-Friendly Rate Control (TFRC) [RFC5348]).

Independent latency estimates SHOULD be maintained for each destination with which an endpoint communicates.

Latency samples MUST NOT be derived from ambiguous transactions. The canonical example is in a protocol that retransmits data, but subsequently cannot determine which copy is being acknowledged. This ambiguity makes correct computation of the latency problematic. See the discussion of Karn's algorithm in [RFC6298]. This requirement ensures a sender establishes a sound estimate of the latency without relying on mis-leading measurements.

When a latency estimate is used to arm a timer that provides loss detection - with or without retransmission - expiry of the timer MUST be interpreted as an indication of congestion in the network, causing the sending rate to be adapted to a safe conservative rate (e.g., TCP collapses the congestion window to one segment [RFC5681]).

Some applications require an initial latency estimate before the latency between endpoints can be empirically sampled. For instance, when arming a retransmission timer an initial value is needed to protect the messages sent before the endpoints sample the latency. This initial latency estimate SHOULD generally be as conservative (large) as possible for the given application. For instance, in the absence of any knowledge about the latency of a path, TCP requires the initial Retransmission Timeout (RTO) to be set to no less than 1 second [RFC6298]. UDP applications SHOULD similarly use an initial latency estimate of 1 second. Values shorter than 1 second can be problematic (see the data analysis in the appendix of [RFC6298]).

3.1.2. Bulk Transfer Applications

Applications that perform bulk transmission of data to a peer over UDP, i.e., applications that exchange more than a few UDP datagrams per round-trip time (RTT), SHOULD implement TFRC [RFC5348], window-based TCP-like congestion control, or otherwise ensure that the application complies with the congestion control principles.

TFRC has been designed to provide both congestion control and fairness in a way that is compatible with the IETF's other transport protocols. If an application implements TFRC, it need not follow the remaining guidelines in Section 3.1.2, because TFRC already addresses them, but SHOULD still follow the remaining guidelines in the subsequent subsections of Section 3.

Bulk transfer applications that choose not to implement TFRC or TCP-like windowing SHOULD implement a congestion control scheme that results in bandwidth (capacity) use that competes fairly with TCP within an order of magnitude.

Section 2 of [RFC3551] suggests that applications SHOULD monitor the packet loss rate to ensure that it is within acceptable parameters.

Packet loss is considered acceptable if a TCP flow across the same network path under the same network conditions would achieve an average throughput, measured on a reasonable timescale, that is not less than that of the UDP flow. The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in timescale and throughput. The recommendations for managing timers specified in Section 3.1.1 also apply.

Finally, some bulk transfer applications may choose not to implement any congestion control mechanism and instead rely on transmitting across reserved path capacity (see Section 3.1.9). This might be an acceptable choice for a subset of restricted networking environments, but is by no means a safe practice for operation over the wider Internet. When the UDP traffic of such applications leaks out into unprovisioned Internet paths, it can significantly degrade the performance of other traffic sharing the path and even result in congestion collapse. Applications that support an uncontrolled or unadaptive transmission behavior **SHOULD NOT** do so by default and **SHOULD** instead require users to explicitly enable this mode of operation, and they **SHOULD** verify that sufficient path capacity has been reserved for them.

3.1.3. Low Data-Volume Applications

When applications that at any time exchange only a few UDP datagrams with a destination implement TFRC or one of the other congestion control schemes in Section 3.1.2, the network sees little benefit, because those mechanisms perform congestion control in a way that is only effective for longer transmissions.

Applications that at any time exchange only a few UDP datagrams with a destination **SHOULD** still control their transmission behavior by not sending on average more than one UDP datagram per RTT to a destination. Similar to the recommendation in [RFC1536], an application **SHOULD** maintain an estimate of the RTT for any destination with which it communicates using the methods specified in Section 3.1.1.

Some applications cannot maintain a reliable RTT estimate for a destination. These applications do not need to or are unable to use protocol timers to measure the RTT (Section 3.1.1). Two cases can be identified:

1. The first case is that of applications that exchange too few UDP datagrams with a peer to establish a statistically accurate RTT estimate, but can monitor the reliability of transmission (Section 3.3). Such applications **MAY** use a predetermined transmission interval that is exponentially backed-off when

packets are found to be lost. TCP specifies an initial value of 1 second [RFC6298], which is also RECOMMENDED as an initial value for UDP applications. Some low data-volume applications, e.g., SIP [RFC3261] and GIST [RFC5971] use an interval of 500 ms, and shorter values are likely problematic in many cases. As in the previous case, note that the initial timeout is not the maximum possible timeout, see Section 3.1.1.

2. A second case of applications cannot maintain an RTT estimate for a destination, because the destination does not send return traffic. Such applications SHOULD NOT send more than one UDP datagram every 3 seconds, and SHOULD use an even less aggressive rate when possible. Shorter values are likely problematic in many cases. Note that the sending rate in this case must be more conservative than in the previous cases, because the lack of return traffic prevents the detection of packet loss, i.e., congestion, and the application therefore cannot perform exponential back-off to reduce load.

3.1.4. Applications supporting bidirectional communications

Applications that communicate bidirectionally SHOULD employ congestion control for both directions of the communication. For example, for a client-server, request-response-style application, clients SHOULD congestion-control their request transmission to a server, and the server SHOULD congestion-control its responses to the clients. Congestion in the forward and reverse direction is uncorrelated, and an application SHOULD either independently detect and respond to congestion along both directions, or limit new and retransmitted requests based on acknowledged responses across the entire round-trip path.

3.1.5. Implications of RTT and Loss Measurements on Congestion Control

Transports such as TCP, SCTP and DCCP provide timely detection of congestion that results in an immediate reduction of their maximum sending rate when congestion is experienced. This reaction is typically completed 1-2 RTTs after loss/congestion is encountered. Applications using UDP SHOULD implement a congestion control scheme that provides a prompt reaction to signals indicating congestion (e.g., by reducing the rate within the next RTT following a congestion signal).

The operation of a UDP congestion control algorithm can be very different to the way TCP operates. This includes congestion controls that respond on timescales that fit applications that cannot usefully work within the "change rate every RTT" model of TCP. Applications that experience a low or varying RTT are particularly vulnerable to

sampling errors (e.g., due to measurement noise, or timer accuracy). This suggests the need to average loss/congestion and RTT measurements over a longer interval, however this also can contribute additional delay in detecting congestion. Some applications may not react by reducing their sending rate immediately for various reasons, including: RTT and loss measurements are only made periodically (e.g., using RTCP), additional time is required to filter information, or the application is only able to change its sending rate at predetermined interval (e.g., some video codecs).

When designing a congestion control algorithm, the designer therefore needs to consider the total time taken to reduce the load following a lack of feedback or a congestion event. An application where the most recent RTT measurement is smaller than the actual RTT or the measured loss rate is smaller than the current rate, can result in over estimating the available capacity. Such over estimation can result in a sending rate that creates congestion to the application or other flows sharing the path capacity, and can contribute to congestion collapse - both of these need to be avoided.

A congestion control designed for UDP SHOULD respond as quickly as possible when it experiences congestion, and SHOULD take into account both the loss rate and the response time when choosing a new rate. The implemented congestion control scheme SHOULD result in bandwidth (capacity) use that is comparable to that of TCP within an order of magnitude, so that it does not starve other flows sharing a common bottleneck.

3.1.6. Burst Mitigation and Pacing

UDP applications SHOULD provide mechanisms to regulate the bursts of transmission that the application may send to the network. Many TCP and SCTP implementations provide mechanisms that prevent a sender from generating long bursts at line-rate, since these are known to induce early loss to applications sharing a common network bottleneck. The use of pacing with TCP [ALLMAN] has also been shown to improve the coexistence of TCP flows with other flows. The need to avoid excessive transmission bursts is also noted in specifications for applications (e.g., [RFC7143]).

Even low data-volume UDP flows may benefit from packet pacing, e.g., an application that sends three copies of a packet to improve robustness to loss is RECOMMENDED to pace out those three packets over several RTTs, to reduce the probability that all three packets will be lost due to the same congestion event (or other event, such as burst corruption).

3.1.7. Explicit Congestion Notification

Internet applications can use Explicit Congestion Notification (ECN) [RFC3168] to gain benefits for the services they support [I-D.ietf-aqm-ecn-benefits].

Internet transports, such as TCP, provide a set of mechanisms that are needed to utilize ECN. ECN operates by setting an ECN-capable codepoint (ECT(0) or ECT(1)) in the IP header of packets that are sent. This indicates to ECN-capable network devices (routers, and other devices) that they may mark (set the congestion experienced, CE codepoint), rather than drop the IP packet as a signal of incipient congestion.

UDP applications can also benefit from enabling ECN, providing that the API supports ECN and that they implement the required protocol mechanisms to support ECN.

The set of mechanisms required for an application to use ECN over UDP are:

- o A sender MUST provide a method to determine (e.g., negotiate) that the corresponding application is able to provide ECN feedback using a compatible ECN method.
- o A receiver that enables the use of ECN for a UDP port MUST check the ECN field at the receiver for each UDP datagram that it receives on this port.
- o The receiving application needs to provide feedback of congestion information to the sending application. This MUST report the presence of datagrams received with a CE-mark by providing a mechanism to feed this congestion information back to the sending application. The feedback MAY also report the presence of ECT(1) and ECT(0)/Not-ECT packets [RFC7560]. ([RFC3168] and [RFC7560] specify methods for TCP.)
- o An application sending ECN-capable datagrams MUST provide an appropriate congestion reaction when it receives feedback indicating that congestion has been experienced. This ought to result in reduction of the sending rate by the UDP congestion control method (see Section 3.1) that is not less than the reaction of TCP under equivalent conditions.
- o A sender SHOULD detect network paths that do not support the ECN field correctly. When detected they need to either conservatively react to congestion or even fall back to not using ECN [I-D.ietf-aqm-ecn-benefits]. This method needs to be robust to

changes within the network path that may occur over the lifetime of a session.

- o A sender is encouraged to provide a mechanism to detect and react appropriately to misbehaving receivers that fail to report CE-marked packets [I-D.ietf-aqm-ecn-benefits].

[RFC6679] provides guidance and an example of this support, by describing a method to allow ECN to be used for UDP-based applications using the Real-Time Protocol (RTP). Applications that cannot provide this set of mechanisms, but wish to gain the benefits of using ECN, are encouraged to use a transport protocol that already supports ECN (such as TCP).

3.1.8. Differentiated Services Model

An application using UDP can use the differentiated services (DiffServ) Quality of Service (QoS) framework. To enable differentiated services processing, a UDP sender sets the Differentiated Services Code Point (DSCP) field [RFC2475] in packets sent to the network. Normally, a UDP source/destination port pair will set a single DSCP value for all packets belonging to a flow, but multiple DSCPs can be used as described later in this section. A DSCP may be chosen from a small set of fixed values (the class selector code points), or from a set of recommended values defined in the Per Hop Behavior (PHB) specifications, or from values that have purely local meanings to a specific network that supports DiffServ. In general, packets may be forwarded across multiple networks between source and destination.

In setting a non-default DSCP value, an application must be aware that DSCP markings may be changed or removed between the traffic source and destination. This has implications on the design of applications that use DSCPs. Specifically, applications SHOULD be designed to not rely on implementation of a specific network treatment, they need instead to implement congestion control methods to determine if their current sending rate is inducing congestion in the network.

[RFC7657] describes the implications of using DSCPs and provides recommendations on using multiple DSCPs within a single network five-tuple (source and destination addresses, source and destination ports, and the transport protocol used, in this case, UDP or UDP-Lite), and particularly the expected impact on transport protocol interactions, with congestion control or reliability functionality (e.g., retransmission, reordering). Use of multiple DSCPs can result in reordering by increasing the set of network forwarding resources

used by a sender. It can also increase exposure to resource depletion or failure.

3.1.9. QoS, Pre-Provisioned or Reserved Capacity

The IETF usually specifies protocols for use within the Best Effort General Internet. Sometimes it is relevant to specify protocols with a different applicability. An application using UDP can use the integrated services QoS framework. This framework is usually made available within controlled environments (e.g., within a single administrative domain or bilaterally agreed connection between domains). Applications intended for the Internet SHOULD NOT assume that QoS mechanisms are supported by the networks they use, and therefore need to provide congestion control, error recovery, etc. in case the actual network path does not provide provisioned service.

Some UDP applications are only expected to be deployed over network paths that use pre-provisioned capacity or capacity reserved using dynamic provisioning, e.g., through the Resource Reservation Protocol (RSVP). Multicast applications are also used with pre-provisioned capacity (e.g., IPTV deployments within access networks). These applications MAY choose not to implement any congestion control mechanism and instead rely on transmitting only on paths where the capacity is provisioned and reserved for this use. This might be an acceptable choice for a subset of restricted networking environments, but is by no means a safe practice for operation over the wider Internet. Applications that choose this option SHOULD carefully and in detail describe the provisioning and management procedures that result in the desired containment.

Applications that support an uncontrolled or unadaptive transmission behavior SHOULD NOT do so by default and SHOULD instead require users to explicitly enable this mode of operation.

Applications designed for use within a controlled environment (see Section 3.6) may be able to exploit network management functions to detect whether they are causing congestion, and react accordingly. If the traffic of such applications leaks out into unprovisioned Internet paths, it can significantly degrade the performance of other traffic sharing the path and even result in congestion collapse. Protocols designed for such networks SHOULD provide mechanisms at the network edge to prevent leakage of traffic into unprovisioned Internet paths (e.g., [RFC7510]). To protect other applications sharing the same path, applications SHOULD also deploy an appropriate circuit breaker, as described in Section 3.1.10.

An IETF specification targeting a controlled environment is expected to provide an applicability statement that restricts the application to the controlled environment (see Section 3.6).

3.1.10. Circuit Breaker Mechanisms

A transport circuit breaker is an automatic mechanism that is used to estimate the congestion caused by a flow, and to terminate (or significantly reduce the rate of) the flow when excessive congestion is detected [I-D.ietf-tsvwg-circuit-breaker]. This is a safety measure to prevent congestion collapse (starvation of resources available to other flows), essential for an Internet that is heterogeneous and for traffic that is hard to predict in advance.

A circuit breaker is intended as a protection mechanism of last resort. Under normal circumstances, a circuit breaker should not be triggered; it is designed to protect things when there is severe overload. The goal is usually to limit the maximum transmission rate that reflects the available capacity of a network path. Circuit breakers can operate on individual UDP flows or traffic aggregates, e.g., traffic sent using a network tunnel.

[I-D.ietf-tsvwg-circuit-breaker] provides guidance and examples on the use of circuit breakers. The use of a circuit breaker in RTP is specified in [I-D.ietf-avtcore-rtp-circuit-breakers].

Applications used in the general Internet SHOULD implement a transport circuit breaker if they do not implement congestion control or operate a low volume data service (see Section 3.6). All applications MAY implement a transport circuit breaker [I-D.ietf-tsvwg-circuit-breaker] and are encouraged to consider implementing at least a slow-acting transport circuit breaker to provide a protection of last resort for their network traffic.

3.1.11. UDP Tunnels

One increasingly popular use of UDP is as a tunneling protocol [I-D.ietf-intarea-tunnels], where a tunnel endpoint encapsulates the packets of another protocol inside UDP datagrams and transmits them to another tunnel endpoint, which decapsulates the UDP datagrams and forwards the original packets contained in the payload. One example of such a protocol is Teredo [RFC4380]. Tunnels establish virtual links that appear to directly connect locations that are distant in the physical Internet topology and can be used to create virtual (private) networks. Using UDP as a tunneling protocol is attractive when the payload protocol is not supported by middleboxes that may exist along the path, because many middleboxes support transmission using UDP.

Well-implemented tunnels are generally invisible to the endpoints that happen to transmit over a path that includes tunneled links. On the other hand, to the routers along the path of a UDP tunnel, i.e., the routers between the two tunnel endpoints, the traffic that a UDP tunnel generates is a regular UDP flow, and the encapsulator and decapsulator appear as regular UDP-sending and -receiving applications. Because other flows can share the path with one or more UDP tunnels, congestion control needs to be considered.

Two factors determine whether a UDP tunnel needs to employ specific congestion control mechanisms -- first, whether the payload traffic is IP-based; second, whether the tunneling scheme generates UDP traffic at a volume that corresponds to the volume of payload traffic carried within the tunnel.

IP-based unicast traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based unicast traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. Consequently, a tunnel carrying IP-based unicast traffic should already interact appropriately with other traffic sharing the path, and specific congestion control mechanisms for the tunnel are not necessary.

However, if the IP traffic in the tunnel is known to not be congestion-controlled, additional measures are RECOMMENDED to limit the impact of the tunneled traffic on other traffic sharing the path. For the specific case of a tunnel that carries IP multicast traffic, see Section 4.1.

The following guidelines define these possible cases in more detail:

1. A tunnel generates UDP traffic at a volume that corresponds to the volume of payload traffic, and the payload traffic is IP-based and congestion-controlled.

This is arguably the most common case for Internet tunnels. In this case, the UDP tunnel SHOULD NOT employ its own congestion control mechanism, because congestion losses of tunneled traffic will already trigger an appropriate congestion response at the original senders of the tunneled traffic. A circuit breaker mechanism may provide benefit by controlling the envelope of the aggregated traffic.

Note that this guideline is built on the assumption that most IP-based communication is congestion-controlled. If a UDP tunnel is used for IP-based traffic that is known to not be congestion-controlled, the next set of guidelines applies.

2. A tunnel generates UDP traffic at a volume that corresponds to the volume of payload traffic, and the payload traffic is not known to be IP-based, or is known to be IP-based but not congestion-controlled.

This can be the case, for example, when some link-layer protocols are encapsulated within UDP (but not all link-layer protocols; some are congestion-controlled). Because it is not known that congestion losses of tunneled non-IP traffic will trigger an appropriate congestion response at the senders, the UDP tunnel SHOULD employ an appropriate congestion control mechanism or circuit breaker mechanism designed for the traffic it carries. Because tunnels are usually bulk-transfer applications as far as the intermediate routers are concerned, the guidelines in Section 3.1.2 apply.

3. A tunnel generates UDP traffic at a volume that does not correspond to the volume of payload traffic, independent of whether the payload traffic is IP-based or congestion-controlled.

Examples of this class include UDP tunnels that send at a constant rate, increase their transmission rates under loss, for example, due to increasing redundancy when Forward Error Correction is used, or are otherwise unconstrained in their transmission behavior. These specialized uses of UDP for tunneling go beyond the scope of the general guidelines given in this document. The implementer of such specialized tunnels SHOULD carefully consider congestion control in the design of their tunneling mechanism and SHOULD consider use of a circuit breaker mechanism.

The type of encapsulated payload might be identified by a UDP port; identified by an Ethernet Type or IP protocol number. A tunnel SHOULD provide mechanisms to restrict the types of flows that may be carried by the tunnel. For instance, a UDP tunnel designed to carry IP needs to filter out non-IP traffic at the ingress. This is particularly important when a generic tunnel encapsulation is used (e.g., one that encapsulates using an EtherType value). Such tunnels SHOULD provide a mechanism to restrict the types of traffic that are allowed to be encapsulated for a given deployment (see [I-D.ietf-intarea-tunnels]).

Designing a tunneling mechanism requires significantly more expertise than needed for many other UDP applications, because tunnels are usually intended to be transparent to the endpoints transmitting over them, so they need to correctly emulate the behavior of an IP link [I-D.ietf-intarea-tunnels], e.g.:

- o Requirements for tunnels that carry or encapsulate using ECN code points [RFC6040].
- o Usage of the IP DSCP field by tunnel endpoints [RFC2983].
- o Encapsulation considerations in the design of tunnels [I-D.ietf-rtgwg-dt-encap].
- o Usage of ICMP messages [I-D.ietf-intarea-tunnels].
- o Handling of fragmentation and packet size for tunnels [I-D.ietf-intarea-tunnels].
- o Source port usage for tunnels designed to support equal cost multipath (ECMP) routing (see Section 5.1.1).
- o Guidance on the need to protect headers [I-D.ietf-intarea-tunnels] and the use of checksums for IPv6 tunnels (see Section 3.4.1).
- o Support for operations and maintenance [I-D.ietf-intarea-tunnels].

At the same time, the tunneled traffic is application traffic like any other from the perspective of the networks the tunnel transmits over. This document only touches upon the congestion control considerations for implementing UDP tunnels; a discussion of other required tunneling behavior is out of scope.

3.2. Message Size Guidelines

IP fragmentation lowers the efficiency and reliability of Internet communication. The loss of a single fragment results in the loss of an entire fragmented packet, because even if all other fragments are received correctly, the original packet cannot be reassembled and delivered. This fundamental issue with fragmentation exists for both IPv4 and IPv6.

In addition, some network address translators (NATs) and firewalls drop IP fragments. The network address translation performed by a NAT only operates on complete IP packets, and some firewall policies also require inspection of complete IP packets. Even with these being the case, some NATs and firewalls simply do not implement the necessary reassembly functionality, and instead choose to drop all fragments. Finally, [RFC4963] documents other issues specific to IPv4 fragmentation.

Due to these issues, an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmission Unit (MTU) along the path to the destination. Consequently, an

application SHOULD either use the path MTU information provided by the IP layer or implement Path MTU Discovery (PMTUD) itself [RFC1191][RFC1981][RFC4821] to determine whether the path to a destination will support its desired message size without fragmentation. However, the ICMP messages that enable path MTU discovery are being increasingly filtered by middleboxes (including Firewalls) [RFC4890]. When the path includes a tunnel, some devices acting as a tunnel ingress discard ICMP messages that originate from network devices over which the tunnel passes, preventing these reaching the UDP endpoint.

Packetization Layer Path MTU Discovery (PLPMTUD) [RFC4821] does not rely upon network support for ICMP messages and is therefore considered more robust than standard PMTUD. It is not susceptible to "black holing" of ICMP message. To operate, PLPMTUD requires changes to the way the transport is used, both to transmit probe packets, and to account for the loss or success of these probes. This updates not only the PMTU algorithm, it also impacts loss recovery, congestion control, etc. These updated mechanisms can be implemented within a connection-oriented transport (e.g., TCP, SCTP, DCCP), but are not a part of UDP, but this type of feedback is not typically present for unidirectional applications.

PLPMTUD therefore places additional design requirements on a UDP application that wishes to use this method. This is especially true for UDP tunnels, because the overhead of sending probe packets needs to be accounted for and may require adding a congestion control mechanism to the tunnel (see Section 3.1.11) as well as complicating the data path at a tunnel decapsulator.

Applications that do not follow this recommendation to do PMTU/PLPMTUD discovery SHOULD still avoid sending UDP datagrams that would result in IP packets that exceed the path MTU. Because the actual path MTU is unknown, such applications SHOULD fall back to sending messages that are shorter than the default effective MTU for sending (EMTU_S in [RFC1122]). For IPv4, EMTU_S is the smaller of 576 bytes and the first-hop MTU [RFC1122]. For IPv6, EMTU_S is 1280 bytes [RFC2460]. The effective PMTU for a directly connected destination (with no routers on the path) is the configured interface MTU, which could be less than the maximum link payload size. Transmission of minimum-sized UDP datagrams is inefficient over paths that support a larger PMTU, which is a second reason to implement PMTU discovery.

To determine an appropriate UDP payload size, applications MUST subtract the size of the IP header (which includes any IPv4 optional headers or IPv6 extension headers) as well as the length of the UDP header (8 bytes) from the PMTU size. This size, known as the Maximum Segment Size (MSS), can be obtained from the TCP/IP stack [RFC1122].

Applications that do not send messages that exceed the effective PMTU of IPv4 or IPv6 need not implement any of the above mechanisms. Note that the presence of tunnels can cause an additional reduction of the effective PMTU [I-D.ietf-intarea-tunnels], so implementing PMTU discovery may be beneficial.

Applications that fragment an application-layer message into multiple UDP datagrams SHOULD perform this fragmentation so that each datagram can be received independently, and be independently retransmitted in the case where an application implements its own reliability mechanisms.

3.3. Reliability Guidelines

Application designers are generally aware that UDP does not provide any reliability, e.g., it does not retransmit any lost packets. Often, this is a main reason to consider UDP as a transport protocol. Applications that do require reliable message delivery MUST implement an appropriate mechanism themselves.

UDP also does not protect against datagram duplication, i.e., an application may receive multiple copies of the same UDP datagram, with some duplicates arriving potentially much later than the first. Application designers SHOULD handle such datagram duplication gracefully, and may consequently need to implement mechanisms to detect duplicates. Even if UDP datagram reception triggers only idempotent operations, applications may want to suppress duplicate datagrams to reduce load.

Applications that require ordered delivery MUST reestablish datagram ordering themselves. The Internet can significantly delay some packets with respect to others, e.g., due to routing transients, intermittent connectivity, or mobility. This can cause reordering, where UDP datagrams arrive at the receiver in an order different from the transmission order.

Applications that use multiple transport ports need to be robust to reordering between sessions. Load-balancing techniques within the network, such as Equal Cost Multipath (ECMP) forwarding can also result in a lack of ordering between different transport sessions, even between the same two network endpoints.

It is important to note that the time by which packets are reordered or after which duplicates can still arrive can be very large. Even more importantly, there is no well-defined upper boundary here. [RFC0793] defines the maximum delay a TCP segment should experience -- the Maximum Segment Lifetime (MSL) -- as 2 minutes. No other RFC defines an MSL for other transport protocols or IP itself. The MSL

value defined for TCP is conservative enough that it SHOULD be used by other protocols, including UDP. Therefore, applications SHOULD be robust to the reception of delayed or duplicate packets that are received within this 2-minute interval.

Retransmission of lost packets or messages is a common reliability mechanism. Such retransmissions can increase network load in response to congestion, worsening that congestion. Any application that uses retransmission is responsible for congestion control of its retransmissions (as well as the application's original traffic), and hence is subject to the Congestion Control guidelines in Section 3.1. Guidance on the appropriate measurement of RTT in Section 3.1.1 also applies for timers used for retransmission packet loss detection.

Instead of implementing these relatively complex reliability mechanisms by itself, an application that requires reliable and ordered message delivery SHOULD whenever possible choose an IETF standard transport protocol that provides these features.

3.4. Checksum Guidelines

The UDP header includes an optional, 16-bit one's complement checksum that provides an integrity check. These checks are not strong from a coding or cryptographic perspective, and are not designed to detect physical-layer errors or malicious modification of the datagram [RFC3819]. Application developers SHOULD implement additional checks where data integrity is important, e.g., through a Cyclic Redundancy Check (CRC) or keyed or non-keyed cryptographic hash included with the data to verify the integrity of an entire object/file sent over the UDP service.

The UDP checksum provides a statistical guarantee that the payload was not corrupted in transit. It also allows the receiver to verify that it was the intended destination of the packet, because it covers the IP addresses, port numbers, and protocol number, and it verifies that the packet is not truncated or padded, because it covers the size field. It therefore protects an application against receiving corrupted payload data in place of, or in addition to, the data that was sent. More description of the set of checks performed using the checksum field is provided in Section 3.1 of [RFC6396].

Applications SHOULD enable UDP checksums [RFC1122]. For IPv4, [RFC0768] permits an option to disable their use, by setting a zero checksum value. An application is permitted to optionally discard UDP datagrams with a zero checksum [RFC1122].

When UDP is used over IPv6, the UDP checksum is relied upon to protect both the IPv6 and UDP headers from corruption (because IPv6

lacks a checksum) and MUST be used as specified in [RFC2460]. Under specific conditions a UDP application is allowed to use a zero UDP zero-checksum mode with a tunnel protocol (see Section 3.4.1).

Applications that choose to disable UDP checksums MUST NOT make assumptions regarding the correctness of received data and MUST behave correctly when a UDP datagram is received that was originally sent to a different destination or is otherwise corrupted.

3.4.1. IPv6 Zero UDP Checksum

[RFC6935] defines a method that enables use of a zero UDP zero-checksum mode with a tunnel protocol, providing that the method satisfies the requirements in [RFC6936]. The application MUST implement mechanisms and/or usage restrictions when enabling this mode. This includes defining the scope for usage and measures to prevent leakage of traffic to other UDP applications (see Appendix A Section 3.6). These additional design requirements for using a zero IPv6 UDP checksum are not present for IPv4, since the IPv4 header validates information that is not protected in an IPv6 packet. Key requirements are:

- o Use of the UDP checksum with IPv6 MUST be the default configuration for all implementations [RFC6935]. The receiving endpoint MUST only allow the use of UDP zero-checksum mode for IPv6 on a UDP destination port that is specifically enabled.
- o An application that support a checksum different to that in [RFC2460] MUST comply with all implementation requirements specified in Section 4 of [RFC6936] and with the usage requirements specified in Section 5 of [RFC6936].
- o A UDP application MUST check that the source and destination IPv6 addresses are valid for any packets with a UDP zero-checksum and MUST discard any packet for which this check fails. To protect from misdelivery, new encapsulation designs SHOULD include an integrity check at the transport layer that includes at least the IPv6 header, the UDP header and the shim header for the encapsulation, if any [RFC6936].
- o One way to help satisfy the requirements of [RFC6936] may be to limit the usage of such tunnels, e.g., to constrain traffic to an operator network, as discussed in Section 3.6. The encapsulation defined for MPLS in UDP [RFC7510] chooses this approach.

As in IPv4, IPv6 applications that choose to disable UDP checksums MUST NOT make assumptions regarding the correctness of received data

and MUST behave correctly when a UDP datagram is received that was originally sent to a different destination or is otherwise corrupted.

IPv6 datagrams with a zero UDP checksum will not be passed by any middlebox that validates the checksum based on [RFC2460] or that updates the UDP checksum field, such as NATs or firewalls. Changing this behavior would require such middleboxes to be updated to correctly handle datagrams with zero UDP checksums. To ensure end-to-end robustness, applications that may be deployed in the general Internet MUST provide a mechanism to safely fall back to using a checksum when a path change occurs that redirects a zero UDP checksum flow over a path that includes a middlebox that discards IPv6 datagrams with a zero UDP checksum.

3.4.2. UDP-Lite

A special class of applications can derive benefit from having partially-damaged payloads delivered, rather than discarded, when using paths that include error-prone links. Such applications can tolerate payload corruption and MAY choose to use the Lightweight User Datagram Protocol (UDP-Lite) [RFC3828] variant of UDP instead of basic UDP. Applications that choose to use UDP-Lite instead of UDP should still follow the congestion control and other guidelines described for use with UDP in Section 3.

UDP-Lite changes the semantics of the UDP "payload length" field to that of a "checksum coverage length" field. Otherwise, UDP-Lite is semantically identical to UDP. The interface of UDP-Lite differs from that of UDP by the addition of a single (socket) option that communicates the checksum coverage length: at the sender, this specifies the intended checksum coverage, with the remaining unprotected part of the payload called the "error-insensitive part." By default, the UDP-Lite checksum coverage extends across the entire datagram. If required, an application may dynamically modify this length value, e.g., to offer greater protection to some messages. UDP-Lite always verifies that a packet was delivered to the intended destination, i.e., always verifies the header fields. Errors in the insensitive part will not cause a UDP datagram to be discarded by the destination. Applications using UDP-Lite therefore MUST NOT make assumptions regarding the correctness of the data received in the insensitive part of the UDP-Lite payload.

A UDP-Lite sender SHOULD select the minimum checksum coverage to include all sensitive payload information. For example, applications that use the Real-Time Protocol (RTP) [RFC3550] will likely want to protect the RTP header against corruption. Applications, where appropriate, MUST also introduce their own appropriate validity

checks for protocol information carried in the insensitive part of the UDP-Lite payload (e.g., internal CRCs).

A UDP-Lite receiver **MUST** set a minimum coverage threshold for incoming packets that is not smaller than the smallest coverage used by the sender [RFC3828]. The receiver **SHOULD** select a threshold that is sufficiently large to block packets with an inappropriately short coverage field. This may be a fixed value, or may be negotiated by an application. UDP-Lite does not provide mechanisms to negotiate the checksum coverage between the sender and receiver. This therefore needs to be performed by the application.

Applications can still experience packet loss when using UDP-Lite. The enhancements offered by UDP-Lite rely upon a link being able to intercept the UDP-Lite header to correctly identify the partial coverage required. When tunnels and/or encryption are used, this can result in UDP-Lite datagrams being treated the same as UDP datagrams, i.e., result in packet loss. Use of IP fragmentation can also prevent special treatment for UDP-Lite datagrams, and this is another reason why applications **SHOULD** avoid IP fragmentation (Section 3.2).

UDP-Lite is supported in some endpoint protocol stacks. Current support for middlebox traversal using UDP-Lite is poor, because UDP-Lite uses a different IPv4 protocol number or IPv6 "next header" value than that used for UDP; therefore, few middleboxes are currently able to interpret UDP-Lite and take appropriate actions when forwarding the packet. This makes UDP-Lite less suited for applications needing general Internet support, until such time as UDP-Lite has achieved better support in middleboxes.

3.5. Middlebox Traversal Guidelines

Network address translators (NATs) and firewalls are examples of intermediary devices ("middleboxes") that can exist along an end-to-end path. A middlebox typically performs a function that requires it to maintain per-flow state. For connection-oriented protocols, such as TCP, middleboxes snoop and parse the connection-management information, and create and destroy per-flow state accordingly. For a connectionless protocol such as UDP, this approach is not possible. Consequently, middleboxes can create per-flow state when they see a packet that -- according to some local criteria -- indicates a new flow, and destroy the state after some time during which no packets belonging to the same flow have arrived.

Depending on the specific function that the middlebox performs, this behavior can introduce a time-dependency that restricts the kinds of UDP traffic exchanges that will be successful across the middlebox. For example, NATs and firewalls typically define the partial path on

one side of them to be interior to the domain they serve, whereas the partial path on their other side is defined to be exterior to that domain. Per-flow state is typically created when the first packet crosses from the interior to the exterior, and while the state is present, NATs and firewalls will forward return traffic. Return traffic that arrives after the per-flow state has timed out is dropped, as is other traffic that arrives from the exterior.

Many applications that use UDP for communication operate across middleboxes without needing to employ additional mechanisms. One example is the Domain Name System (DNS), which has a strict request-response communication pattern that typically completes within seconds.

Other applications may experience communication failures when middleboxes destroy the per-flow state associated with an application session during periods when the application does not exchange any UDP traffic. Applications SHOULD be able to gracefully handle such communication failures and implement mechanisms to re-establish application-layer sessions and state.

For some applications, such as media transmissions, this re-synchronization is highly undesirable, because it can cause user-perceivable playback artifacts. Such specialized applications MAY send periodic keep-alive messages to attempt to refresh middlebox state (e.g., [RFC7675]). It is important to note that keep-alive messages are not recommended for general use -- they are unnecessary for many applications and can consume significant amounts of system and network resources.

An application that needs to employ keep-alive messages to deliver useful service over UDP in the presence of middleboxes SHOULD NOT transmit them more frequently than once every 15 seconds and SHOULD use longer intervals when possible. No common timeout has been specified for per-flow UDP state for arbitrary middleboxes. NATs require a state timeout of 2 minutes or longer [RFC4787]. However, empirical evidence suggests that a significant fraction of currently deployed middleboxes unfortunately use shorter timeouts. The timeout of 15 seconds originates with the Interactive Connectivity Establishment (ICE) protocol [RFC5245]. When an application is deployed in a controlled environment, the deployer SHOULD investigate whether the target environment allows applications to use longer intervals, or whether it offers mechanisms to explicitly control middlebox state timeout durations, for example, using the Port Control Protocol (PCP) [RFC6887], Middlebox Communications (MIDCOM) [RFC3303], Next Steps in Signaling (NSIS) [RFC5973], or Universal Plug and Play (UPnP) [UPnP]. It is RECOMMENDED that applications apply slight random variations ("jitter") to the timing of keep-alive

transmissions, to reduce the potential for persistent synchronization between keep-alive transmissions from different hosts [RFC7675].

Sending keep-alive messages is not a substitute for implementing a mechanism to recover from broken sessions. Like all UDP datagrams, keep-alive messages can be delayed or dropped, causing middlebox state to time out. In addition, the congestion control guidelines in Section 3.1 cover all UDP transmissions by an application, including the transmission of middlebox keep-alive messages. Congestion control may thus lead to delays or temporary suspension of keep-alive transmission.

Keep-alive messages are NOT RECOMMENDED for general use. They are unnecessary for many applications and may consume significant resources. For example, on battery-powered devices, if an application needs to maintain connectivity for long periods with little traffic, the frequency at which keep-alive messages are sent can become the determining factor that governs power consumption, depending on the underlying network technology.

Because many middleboxes are designed to require keep-alive messages for TCP connections at a frequency that is much lower than that needed for UDP, this difference alone can often be sufficient to prefer TCP over UDP for these deployments. On the other hand, there is anecdotal evidence that suggests that direct communication through middleboxes, e.g., by using ICE [RFC5245], does succeed less often with TCP than with UDP. The trade-offs between different transport protocols -- especially when it comes to middlebox traversal -- deserve careful analysis.

UDP applications that could be deployed in the Internet need to be designed understanding that there are many variants of middlebox behavior, and although UDP is connectionless, middleboxes often maintain state for each UDP flow. Using multiple UDP flows can consume available state space and also can lead to changes in the way the middlebox handles subsequent packets (either to protect its internal resources, or to prevent perceived misuse). The probability of path failure can increase when applications use multiple UDP flows in parallel (see Section 5.1.2 for recommendations on usage of multiple ports).

3.6. Limited Applicability and Controlled Environments

Two different types of applicability have been identified for the specification of IETF applications that utilize UDP:

General Internet. By default, IETF specifications target deployment on the general Internet. Experience has shown that successful

protocols developed in one specific context or for a particular application tend to become used in a wider range of contexts. For example, a protocol with an initial deployment within a local area network may subsequently be used over a virtual network that traverses the Internet, or in the Internet in general. Applications designed for general Internet use may experience a range of network device behaviors, and in particular should consider whether applications need to operate over paths that may include middleboxes.

Controlled Environment. A protocol/encapsulation/tunnel could be designed to be used only within a controlled environment. For example, an application designed for use by a network operator might only be deployed within the network of that single network operator or on networks of an adjacent set of cooperating network operators. The application traffic may then be managed to avoid congestion, rather than relying on built-in mechanisms, which are required when operating over the general Internet. Applications that target a limited applicability use case may be able to take advantage of specific hardware (e.g., carrier-grade equipment) or underlying protocol features of the subnetwork over which they are used.

Specifications addressing a limited applicability use case or a controlled environment SHOULD identify how in their restricted deployment a level of safety is provided that is equivalent to that of a protocol designed for operation over the general Internet (e.g., a design based on extensive experience with deployments of particular methods that provide features that cannot be expected in general Internet equipment and the robustness of the design of MPLS to corruption of headers both helped justify use of an alternate UDP integrity check [RFC7510]).

An IETF specification targeting a controlled environment is expected to provide an applicability statement that restricts the application traffic to the controlled environment, and would be expected to describe how methods can be provided to discourage or prevent escape of corrupted packets from the environment (for example, section 5 of [RFC7510]).

4. Multicast UDP Usage Guidelines

This section complements Section 3 by providing additional guidelines that are applicable to multicast and broadcast usage of UDP.

Multicast and broadcast transmission [RFC1112] usually employ the UDP transport protocol, although they may be used with other transport protocols (e.g., UDP-Lite).

There are currently two models of multicast delivery: the Any-Source Multicast (ASM) model as defined in [RFC1112] and the Source-Specific Multicast (SSM) model as defined in [RFC4607]. ASM group members will receive all data sent to the group by any source, while SSM constrains the distribution tree to only one single source.

Specialized classes of applications also use UDP for IP multicast or broadcast [RFC0919]. The design of such specialized applications requires expertise that goes beyond simple, unicast-specific guidelines, since these senders may transmit to potentially very many receivers across potentially very heterogeneous paths at the same time, which significantly complicates congestion control, flow control, and reliability mechanisms.

This section provides guidance on multicast and broadcast UDP usage. Use of broadcast by an application is normally constrained by routers to the local subnetwork. However, use of tunneling techniques and proxies can and does result in some broadcast traffic traversing Internet paths. These guidelines therefore also apply to broadcast traffic.

The IETF has defined a reliable multicast framework [RFC3048] and several building blocks to aid the designers of multicast applications, such as [RFC3738] or [RFC4654].

Senders to anycast destinations must be aware that successive messages sent to the same anycast IP address may be delivered to different anycast nodes, i.e., arrive at different locations in the topology.

Most UDP tunnels that carry IP multicast traffic use a tunnel encapsulation with a unicast destination address, such as Automatic Multicast Tunneling [RFC7450]. These MUST follow the same requirements as a tunnel carrying unicast data (see Section 3.1.11). There are deployment cases and solutions where the outer header of a UDP tunnel contains a multicast destination address, such as [RFC6513]. These cases are primarily deployed in controlled environments over reserved capacity, often operating within a single administrative domain, or between two domains over a bi-laterally agreed upon path with reserved capacity, and so congestion control is OPTIONAL, but circuit breaker techniques are still RECOMMENDED in order to restore some degree of service should the offered load exceed the reserved capacity (e.g., due to misconfiguration).

4.1. Multicast Congestion Control Guidelines

Unicast congestion-controlled transport mechanisms are often not applicable to multicast distribution services, or simply do not scale to large multicast trees, since they require bi-directional communication and adapt the sending rate to accommodate the network conditions to a single receiver. In contrast, multicast distribution trees may fan out to massive numbers of receivers, which limits the scalability of an in-band return channel to control the sending rate, and the one-to-many nature of multicast distribution trees prevents adapting the rate to the requirements of an individual receiver. For this reason, generating TCP-compatible aggregate flow rates for Internet multicast data, either native or tunneled, is the responsibility of the application implementing the congestion control.

Applications using multicast SHOULD provide appropriate congestion control. Multicast congestion control needs to be designed using mechanisms that are robust to the potential heterogeneity of both the multicast distribution tree and the receivers belonging to a group. Heterogeneity may manifest itself in some receivers experiencing more loss than others, higher delay, and/or less ability to respond to network conditions. Congestion control is particularly important for any multicast session where all or part of the multicast distribution tree spans an access network (e.g., a home gateway). Two styles of congestion control have been defined in the RFC-series:

- o Feedback-based congestion control, in which the sender receives multicast or unicast UDP messages from the receivers allowing it to assess the level of congestion and then adjust the sender rate(s) (e.g., [RFC5740],[RFC4654]). Multicast methods may operate on longer timescales than for unicast (e.g., due to the higher group RTT of a heterogeneous group). A control method could decide not to reduce the rate of the entire multicast group in response to a control message received from a single receiver (e.g., a sender could set a minimum rate and decide to request a congested receiver to leave the multicast group and could also decide to distribute content to these congested receivers at a lower rate using unicast congestion control).
- o Receiver-driven congestion control, which does not require a receiver to send explicit UDP control messages for congestion control (e.g., [RFC3738], [RFC5775]). Instead, the sender distributes the data across multiple IP multicast groups (e.g., using a set of {S,G} channels). Each receiver determines its own level of congestion and controls its reception rate using only multicast join/leave messages sent in the network control plane. This method scales to arbitrary large groups of receivers.

Any multicast-enabled receiver may attempt to join and receive traffic from any group. This may imply the need for rate limits on individual receivers or the aggregate multicast service. Note there is no way at the transport layer to prevent a join message propagating to the next-hop router.

Some classes of multicast applications support applications that can monitor the user-level quality of the transfer at the receiver. Applications that can detect a significant reduction in user quality SHOULD regard this as a congestion signal (e.g., to leave a group using layered multicast encoding) or, if not, SHOULD use this signal to provide a circuit breaker to terminate the flow by leaving the multicast group.

4.1.1. Bulk Transfer Multicast Applications

Applications that perform bulk transmission of data over a multicast distribution tree, i.e., applications that exchange more than a few UDP datagrams per RTT, SHOULD implement a method for congestion control. The currently RECOMMENDED IETF methods are: Asynchronous Layered Coding (ALC) [RFC5775], TCP-Friendly Multicast Congestion Control (TFMCC) [RFC4654], Wave and Equation Based Rate Control (WEBRC) [RFC3738], NACK-Oriented Reliable Multicast (NORM) transport protocol [RFC5740], File Delivery over Unidirectional Transport (FLUTE) [RFC6726], Real Time Protocol/Control Protocol (RTP/RTCP) [RFC3550].

An application can alternatively implement another congestion control schemes following the guidelines of [RFC2887] and utilizing the framework of [RFC3048]. Bulk transfer applications that choose not to implement [RFC4654], [RFC5775], [RFC3738], [RFC5740], [RFC6726], or [RFC3550] SHOULD implement a congestion control scheme that results in bandwidth use that competes fairly with TCP within an order of magnitude.

Section 2 of [RFC3551] states that multimedia applications SHOULD monitor the packet loss rate to ensure that it is within acceptable parameters. Packet loss is considered acceptable if a TCP flow across the same network path under the same network conditions would achieve an average throughput, measured on a reasonable timescale, that is not less than that of the UDP flow. The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in timescale and throughput.

4.1.2. Low Data-Volume Multicast Applications

All the recommendations in Section 3.1.3 are also applicable to low data-volume multicast applications.

4.2. Message Size Guidelines for Multicast

A multicast application SHOULD NOT send UDP datagrams that result in IP packets that exceed the effective MTU as described in section 3 of [RFC6807]. Consequently, an application SHOULD either use the effective MTU information provided by the Population Count Extensions to Protocol Independent Multicast [RFC6807] or implement path MTU discovery itself (see Section 3.2) to determine whether the path to each destination will support its desired message size without fragmentation.

5. Programming Guidelines

The de facto standard application programming interface (API) for TCP/IP applications is the "sockets" interface [POSIX]. Some platforms also offer applications the ability to directly assemble and transmit IP packets through "raw sockets" or similar facilities. This is a second, more cumbersome method of using UDP. The guidelines in this document cover all such methods through which an application may use UDP. Because the sockets API is by far the most common method, the remainder of this section discusses it in more detail.

Although the sockets API was developed for UNIX in the early 1980s, a wide variety of non-UNIX operating systems also implement it. The sockets API supports both IPv4 and IPv6 [RFC3493]. The UDP sockets API differs from that for TCP in several key ways. Because application programmers are typically more familiar with the TCP sockets API, this section discusses these differences. [STEVENS] provides usage examples of the UDP sockets API.

UDP datagrams may be directly sent and received, without any connection setup. Using the sockets API, applications can receive packets from more than one IP source address on a single UDP socket. Some servers use this to exchange data with more than one remote host through a single UDP socket at the same time. Many applications need to ensure that they receive packets from a particular source address; these applications MUST implement corresponding checks at the application layer or explicitly request that the operating system filter the received packets.

Many operating systems also allow a UDP socket to be connected, i.e., to bind a UDP socket to a specific pair of addresses and ports. This

is similar to the corresponding TCP sockets API functionality. However, for UDP, this is only a local operation that serves to simplify the local send/receive functions and to filter the traffic for the specified addresses and ports. Binding a UDP socket does not establish a connection -- UDP does not notify the remote end when a local UDP socket is bound. Binding a socket also allows configuring options that affect the UDP or IP layers, for example, use of the UDP checksum or the IP Timestamp option. On some stacks, a bound socket also allows an application to be notified when ICMP error messages are received for its transmissions [RFC1122].

If a client/server application executes on a host with more than one IP interface, the application SHOULD send any UDP responses with an IP source address that matches the IP destination address of the UDP datagram that carried the request (see [RFC1122], Section 4.1.3.5). Many middleboxes expect this transmission behavior and drop replies that are sent from a different IP address, as explained in Section 3.5.

A UDP receiver can receive a valid UDP datagram with a zero-length payload. Note that this is different from a return value of zero from a `read()` socket call, which for TCP indicates the end of the connection.

UDP provides no flow-control, i.e., the sender at any given time does not know whether the receiver is able to handle incoming transmissions. This is another reason why UDP-based applications need to be robust in the presence of packet loss. This loss can also occur within the sending host, when an application sends data faster than the line rate of the outbound network interface. It can also occur at the destination, where receive calls fail to return all the data that was sent when the application issues them too infrequently (i.e., such that the receive buffer overflows). Robust flow control mechanisms are difficult to implement, which is why applications that need this functionality SHOULD consider using a full-featured transport protocol such as TCP.

When an application closes a TCP, SCTP or DCCP socket, the transport protocol on the receiving host is required to maintain TIME-WAIT state. This prevents delayed packets from the closed connection instance from being mistakenly associated with a later connection instance that happens to reuse the same IP address and port pairs. The UDP protocol does not implement such a mechanism. Therefore, UDP-based applications need to be robust to reordering and delay. One application may close a socket or terminate, followed in time by another application receiving on the same port. This later application may then receive packets intended for the first application that were delayed in the network.

5.1. Using UDP Ports

The rules and procedures for the management of the Service Name and Transport Protocol Port Number Registry are specified in [RFC6335]. Recommendations for use of UDP ports are provided in [RFC7605].

A UDP sender SHOULD NOT use a source port value of zero. A source port number that cannot be easily determined from the address or payload type provides protection at the receiver from data injection attacks by off-path devices. A UDP receiver SHOULD NOT bind to port zero.

Applications SHOULD implement receiver port and address checks at the application layer or explicitly request that the operating system filter the received packets to prevent receiving packets with an arbitrary port. This measure is designed to provide additional protection from data injection attacks from an off-path source (where the port values may not be known).

Applications SHOULD provide a check that protects from off-path data injection, avoiding an application receiving packets that were created by an unauthorized third party. TCP stacks commonly use a randomized source port to provide this protection [RFC6056]; UDP applications should follow the same technique. Middleboxes and end systems often make assumptions about the system ports or user ports, hence it is recommended to use randomized ports in the Dynamic and/or Private Port range. Setting a "randomized" source port also provides greater assurance that reported ICMP errors originate from network systems on the path used by a particular flow. Some UDP applications choose to use a predetermined value for the source port (including some multicast applications), these applications need to therefore employ a different technique. Protection from off-path data attacks can also be provided by randomizing the initial value of another protocol field within the datagram payload, and checking the validity of this field at the receiver (e.g., RTP has random initial sequence number and random media timestamp offsets [RFC3550]).

When using multicast, IP routers perform a reverse-path forwarding (RPF) check for each multicast packet. This provides protection from off-path data injection. When a receiver joins a multicast group and filters based on the source address the filter verifies the sender's IP address. This is always the case when using a SSM {S,G} channel.

5.1.1. Usage of UDP for source port entropy and the IPv6 Flow Label

Some applications use the UDP datagram header as a source of entropy for network devices that implement ECMP [RFC6438]. A UDP tunnel application targeting this usage, encapsulates an inner packet using

UDP, where the UDP source port value forms a part of the entropy that can be used to balance forwarding of network traffic by the devices that use ECMP. A sending tunnel endpoint selects a source port value in the UDP datagram header that is computed from the inner flow information (e.g., the encapsulated packet headers). To provide sufficient entropy the sending tunnel endpoint maps the encapsulated traffic to one of a range of UDP source values. The value SHOULD be within the ephemeral port range, i.e., 49152 to 65535, where the high order two bits of the port are set to one. The available source port entropy of 14 bits (using the ephemeral port range) plus the outer IP addresses seems sufficient for entropy for most ECMP applications [I-D.ietf-rtgwg-dt-encap].

To avoid reordering within an IP flow, the same UDP source port value SHOULD be used for all packets assigned to an encapsulated flow (e.g., using a hash of the relevant headers). The entropy mapping for a flow MAY change over the lifetime of the encapsulated flow [I-D.ietf-rtgwg-dt-encap]. For instance, this could be changed as a Denial of Service (DOS) mitigation, or as a means to effect routing through the ECMP network. However, the source port selected for a flow SHOULD NOT change more than once in every thirty seconds (e.g., as in [I-D.ietf-tsvwg-gre-in-udp-encap]).

The use of the source port field for entropy has several side-effects that need to be considered, including:

- o 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.
- o 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.
- o 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.

5.1.1.2. 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:

- o Starting a series of successive connections can increase the number of state bindings in middleboxes (e.g., NAT 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.
- o 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).
- o Using several flows can also increase the occupancy of a binding or lookup table in a middlebox (e.g., NAT or Firewall), which may cause the device to change the way it manages the flow state.

- o Further, using excessive numbers of flows can degrade the ability of a unicast congestion control to react to congestion events, unless the congestion state is shared between all flows in a session. A receiver-driven multicast congestion control requires the sending application to distribute its data over a set of IP multicast groups, each receiver is therefore expected to receive data from a modest number of simultaneously active UDP ports.

Therefore, applications MUST NOT assume consistent behavior of middleboxes when multiple UDP flows are used; many devices respond differently as the number of used ports increases. Using multiple flows with different QoS requirements requires applications to verify that the expected performance is achieved using each individual flow (five-tuple), see Section 3.1.9.

5.2. ICMP Guidelines

Applications can utilize information about ICMP error messages that the UDP layer passes up for a variety of purposes [RFC1122]. Applications SHOULD appropriately validate the payload of ICMP messages to ensure these are received in response to transmitted traffic (i.e., a reported error condition that corresponds to a UDP datagram actually sent by the application). This requires context, such as local state about communication instances to each destination, that although readily available in connection-oriented transport protocols is not always maintained by UDP-based applications. Note that not all platforms have the necessary APIs to support this validation, and some platforms already perform this validation internally before passing ICMP information to the application.

Any application response to ICMP error messages SHOULD be robust to temporary routing failures (sometimes called "soft errors"), e.g., transient ICMP "unreachable" messages ought to not normally cause a communication abort.

ICMP messages are being increasingly filtered by middleboxes. A UDP application therefore SHOULD NOT rely on their delivery for correct and safe operation.

6. Security Considerations

UDP does not provide communications security. Applications that need to protect their communications against eavesdropping, tampering, or message forgery SHOULD employ end-to-end security services provided by other IETF protocols.

UDP applications SHOULD provide protection from off-path data injection attacks using a randomized source port or equivalent technique (see Section 5.1).

Applications that respond to short requests with potentially large responses are a potential vector for amplification attacks, and SHOULD take steps to minimize their potential for being abused as part of a DoS attack. That could mean authenticating the sender before responding; noting that the source IP address of a request is not a useful authenticator, because it can easily be spoofed. Or it may mean otherwise limiting the cases where short unauthenticated requests produce large responses. Applications MAY also want to offer ways to limit the number of requests they respond to in a time interval, in order to cap the bandwidth they consume.

One option for securing UDP communications is with IPsec [RFC4301], which can provide authentication for flows of IP packets through the Authentication Header (AH) [RFC4302] and encryption and/or authentication through the Encapsulating Security Payload (ESP) [RFC4303]. Applications use the Internet Key Exchange (IKE) [RFC7296] to configure IPsec for their sessions. Depending on how IPsec is configured for a flow, it can authenticate or encrypt the UDP headers as well as UDP payloads. If an application only requires authentication, ESP with no encryption but with authentication is often a better option than AH, because ESP can operate across middleboxes. An application that uses IPsec requires the support of an operating system that implements the IPsec protocol suite, and the network path must permit IKE and IPsec traffic. This may become more common with IPv6 deployments [RFC6092].

Although it is possible to use IPsec to secure UDP communications, not all operating systems support IPsec or allow applications to easily configure it for their flows. A second option for securing UDP communications is through Datagram Transport Layer Security (DTLS) [RFC6347][RFC7525]. DTLS provides communication privacy by encrypting UDP payloads. It does not protect the UDP headers. Applications can implement DTLS without relying on support from the operating system.

Many other options for authenticating or encrypting UDP payloads exist. For example, the GSS-API security framework [RFC2743] or Cryptographic Message Syntax (CMS) [RFC5652] could be used to protect UDP payloads. There exist a number of security options for RTP [RFC3550] over UDP, especially to accomplish key-management, see [RFC7201]. These options covers many usages, including point-to-point, centralized group communication as well as multicast. In some applications, a better solution is to protect larger stand-alone objects, such as files or messages, instead of individual UDP

payloads. In these situations, CMS [RFC5652], S/MIME [RFC5751] or OpenPGP [RFC4880] could be used. In addition, there are many non-IETF protocols in this area.

Like congestion control mechanisms, security mechanisms are difficult to design and implement correctly. It is hence RECOMMENDED that applications employ well-known standard security mechanisms such as DTLS or IPsec, rather than inventing their own.

The Generalized TTL Security Mechanism (GTSM) [RFC5082] may be used with UDP applications when the intended endpoint is on the same link as the sender. This lightweight mechanism allows a receiver to filter unwanted packets.

In terms of congestion control, [RFC2309] and [RFC2914] discuss the dangers of congestion-unresponsive flows to the Internet. [I-D.ietf-tsvwg-circuit-breaker] describes methods that can be used to set a performance envelope that can assist in preventing congestion collapse in the absence of congestion control or when the congestion control fails to react to congestion events. This document provides guidelines to designers of UDP-based applications to congestion-control their transmissions, and does not raise any additional security concerns.

Some network operators have experienced surges of UDP attack traffic that are multiple orders of magnitude above the baseline traffic rate for UDP. This can motivate operators to limit the data rate or packet rate of UDP traffic. This may in turn limit the throughput that an application can achieve using UDP and could also result in higher packet loss for UDP traffic that would not be experienced if other transport protocols had been used.

A UDP application with a long-lived association between the sender and receiver, ought to be designed so that the sender periodically checks that the receiver still wants ("consents") to receive traffic and need to be designed to stop if there is no explicit confirmation of this [RFC7675]. Applications that require communications in two directions to implement protocol functions (such as reliability or congestion control) will need to independently check both directions of communication, and may have to exchange keep-alive messages to traverse middleboxes (see Section 3.5).

7. Summary

This section summarizes the key guidelines made in Sections 3 - 6 in a tabular format (Table 1) for easy referencing.

+-----+-----+

Recommendation	Section
MUST tolerate a wide range of Internet path conditions 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:

- o Addressed IESG review comments.

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

- o Fix a nit.

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

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

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

- o Addressed suggestions by David Black.

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

- o Addressed more suggestions by Takeshi Takahashi.

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

- o Addressed SEC_DIR review by Takeshi Takahashi.
- o Addressed Gen-ART review by Paul Kyzivat.
- o Addressed OPS-DIR review by Tim Chown.
- o Addressed some of Mark Allman's comments regarding RTTs and RTOs.
- o Addressed some of Brian Trammell's comments regarding new IETF transports.

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

- o Minor corrections.
- o Changes recommended by Spencer Dawkins.
- o Placed the recommendations on timers within section 3.1
- o Updated the recommendations on reliability to also reference the recommendations on timers.

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

- o Introduced a separate section on the usage of timers to avoid repeating similar guidance in multiple sections.
- o Updated RTT measurement text to align with revised min RTO recommendation for TCP.

- o Updated text based on draft-ietf-tcpm-rto-consider-03 and to now cite this draft.
- o Fixed inconsistency in term used for keep-alive messages (keep-alive packet, keep-alives).

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

- o Address some issues that idnits found.

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

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

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

- o Fix to cross reference in summary table.

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

This update introduces new text in the following sections:

- o The ID from RTGWG on encap Section 7 makes recommendations on entropy. Section 5.1 of the 5405bis draft had a single sentence on use of the UDP source port to inject entropy. Related work such as UDP-in-MPLS and GRE-in-UDP have also made recommendations on entropy usage. A new section has been added to address this.
- o Added reference to RFC2983 on DSCP with tunnels.
- o New text after comment from David Black on needing to improve the header protection text.
- o Replaced replace /controlled network environment/ with /controlled environment/ to be more consistent with other drafts.
- o Section 3.1.7 now explicitly refers to the applicability subsection describing controlled environments.
- o PLPMTUD section updated.
- o Reworded checksum text to place IPv6 UDP zero checksum text in a separate subsection (this became too long in the main section)
- o Updated summary table

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

This update introduces new text in the following sections:

- o Addressed David Black's review during WG LC.

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

This update introduces new text in the following sections:

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

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

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

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

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

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

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

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

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

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

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

- o Added text on DSCP-usage.
- o More guidance on use of the checksum, including an example of how MPLS/UDP allowed support of a zero IPv6 UDP Checksum in some cases.
- o Added description of diffuse usage.
- o Clarified usage of the source port field.

draft-eggert-tsvwg-rfc5405bis-01 was adopted by the TSVWG and resubmitted as draft-ietf-tsvwg-rfc5405bis-00. There were no technical changes.

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

- o Added Greg Shepherd as a co-author, based on the multicast guidelines that originated with him.

Changes in draft-eggert-tsvwg-rfc5405bis-00 (relative to RFC5405):

- o The words "application designers" were removed from the draft title and the wording of the abstract was clarified abstract.
- o New text to clarify various issues and set new recommendations not previously included in RFC 5405. These include new recommendations for multicast, the use of checksums with IPv6, ECMP, recommendations on port usage, use of ECN, use of DiffServ, circuit breakers (initial text), etc.

Authors' Addresses

Lars Eggert
NetApp
Sonnenallee 1
Kirchheim 85551
Germany

Phone: +49 151 120 55791
EMail: lars@netapp.com
URI: <https://eggert.org/>

Godred Fairhurst
University of Aberdeen
Department of Engineering
Fraser Noble Building
Aberdeen AB24 3UE
Scotland

EMail: gorry@erg.abdn.ac.uk
URI: <http://www.erg.abdn.ac.uk/>

Greg Shepherd
Cisco Systems
Tasman Drive
San Jose
USA

EMail: gjshep@gmail.com

Transport Area Working Group
Internet-Draft
Updates: 6040, 2661, 2784, 3931, 4380,
7450 (if approved)
Intended status: Standards Track
Expires: November 25, 2021

B. Briscoe
Independent
May 24, 2021

Propagating Explicit Congestion Notification Across IP Tunnel Headers
Separated by a Shim
draft-ietf-tsvwg-rfc6040update-shim-14

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 updates RFC 6040 to clarify that its scope includes tunnels where two IP headers are separated by at least one shim header that is not sufficient on its own for wide area packet forwarding. It surveys widely deployed IP tunnelling protocols that use such shim header(s) and updates the specifications of those that do not mention ECN propagation (L2TPv2, L2TPv3, GRE, Teredo and AMT). This specification also updates RFC 6040 with configuration requirements needed to make any legacy tunnel ingress safe.

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 November 25, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Scope of RFC 6040	3
3.1. Feasibility of ECN Propagation between Tunnel Headers . .	4
3.2. Desirability of ECN Propagation between Tunnel Headers .	5
4. Making a non-ECN Tunnel Ingress Safe by Configuration	5
5. ECN Propagation and Fragmentation/Reassembly	7
6. IP-in-IP Tunnels with Tightly Coupled Shim Headers	7
6.1. Specific Updates to Protocols under IETF Change Control .	10
6.1.1. L2TP (v2 and v3) ECN Extension	10
6.1.2. GRE	13
6.1.3. Teredo	14
6.1.4. AMT	15
7. IANA Considerations	17
8. Security Considerations	17
9. Comments Solicited	17
10. Acknowledgements	17
11. References	18
11.1. Normative References	18
11.2. Informative References	19
Author's Address	22

1. Introduction

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.

A common pattern for many tunnelling protocols is to encapsulate an inner IP header (v4 or v6) with shim header(s) then an outer IP header (v4 or v6). Some of these shim headers are designed as generic encapsulations, so they do not necessarily directly encapsulate an inner IP header. Instead they can encapsulate headers such as link-layer (L2) protocols that in turn often encapsulate IP.

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) and other encapsulations between the IP headers. Where necessary, it updates the specifications of the relevant encapsulation protocols with the specific text necessary to comply with RFC 6040.

This specification also updates RFC 6040 to state how operators ought to configure a legacy tunnel ingress to avoid unsafe system configurations.

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] when, and only when, they appear in all capitals, as shown here.

This specification uses the terminology defined in RFC 6040 [RFC6040].

3. Scope of RFC 6040

In section 1.1 of RFC 6040, its scope is defined 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. ..."

This was intended to include cases where shim header(s) sit between the IP headers. Many tunnelling implementers have interpreted the scope of RFC 6040 as it was intended, but it is ambiguous. Therefore, this specification updates RFC 6040 by adding the following scoping text after the sentences quoted above:

It applies in cases where an outer IP header encapsulates an inner IP header either directly or indirectly by encapsulating other headers that in turn encapsulate (or might encapsulate) an inner IP header.

There is another problem with the scope of RFC 6040. Like many IETF specifications, RFC 6040 is written as a specification that implementations can choose to claim compliance with. This means it does not cover two important cases:

1. those cases where it is infeasible for an implementation to access an inner IP header when adding or removing an outer IP header;
2. those implementations that choose not to propagate ECN between IP headers.

However, the ECN field is a non-optional part of the IP header (v4 and v6). So any implementation that creates an outer IP header has to give the ECN field some value. There is only one safe value a tunnel ingress can use if it does not know whether the egress supports propagation of the ECN field; it has to clear the ECN field in any outer IP header to 0b00.

However, an RFC has no jurisdiction over implementations that choose not to comply with it or cannot comply with it, including all those implementations that pre-dated the RFC. Therefore it would have been unreasonable to add such a requirement to RFC 6040. Nonetheless, to ensure safe propagation of the ECN field over tunnels, it is reasonable to add requirements on operators, to ensure they configure their tunnels safely (where possible). Before stating these configuration requirements in Section 4, the factors that determine whether propagating ECN is feasible or desirable will be briefly introduced.

3.1. Feasibility of ECN Propagation between Tunnel Headers

In many cases shim header(s) and an outer IP header are always added to (or removed from) an inner IP packet as part of the same procedure. 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. In these cases it will often be feasible for an implementation to propagate the ECN field between the IP headers.

In some cases a tunnel adds an outer IP header and a tightly coupled shim header to an inner header that is not an IP header, but that in turn encapsulates an IP header (or might encapsulate an IP header). For instance an inner Ethernet (or other link layer) header might encapsulate an inner IP header as its payload. We call this a tightly coupled shim over an encapsulating header.

Digging to arbitrary depths to find an inner IP header within an encapsulation is strictly a layering violation so it cannot be a required behaviour. Nonetheless, some tunnel endpoints already look within a L2 header for an IP header, for instance to map the Diffserv codepoint between an encapsulated IP header and an outer IP header

[RFC2983]. In such cases at least, it should be feasible to also (independently) propagate the ECN field between the same IP headers. Thus, access to the ECN field within an encapsulating header can be a useful and benign optimization. The guidelines in section 5 of [I-D.ietf-tsvwg-ecn-encap-guidelines] give the conditions for this layering violation to be benign.

3.2. Desirability of ECN Propagation between Tunnel Headers

Developers and network operators are encouraged to implement and deploy tunnel endpoints compliant with RFC 6040 (as updated by the present specification) in order to provide the benefits of wider ECN deployment [RFC8087]. Nonetheless, propagation of ECN between IP headers, whether separated by shim headers or not, has to be optional to implement and to use, because:

- o Legacy implementations of tunnels without any ECN support already exist
- o A network might be designed so that there is usually no bottleneck within the tunnel
- o If the tunnel endpoints would have to search within an L2 header to find an encapsulated IP header, it might not be worth the potential performance hit

4. Making a non-ECN Tunnel Ingress Safe by Configuration

Even when no specific attempt has been made to implement propagation of the ECN field at a tunnel ingress, it ought to be possible for the operator to render a tunnel ingress safe by configuration. The main safety concern is to disable (clear to zero) the ECN capability in the outer IP header at the ingress if the egress of the tunnel does not implement ECN logic to propagate any ECN markings into the packet forwarded beyond the tunnel. Otherwise the non-ECN egress could discard any ECN marking introduced within the tunnel, which would break all the ECN-based control loops that regulate the traffic load over the tunnel.

Therefore this specification updates RFC 6040 by inserting the following text at the end of section 4.3:

"

Whether or not an ingress implementation claims compliance with RFC 6040, RFC 4301 or RFC3168, when the outer tunnel header is IP (v4 or v6), if possible, the operator MUST configure the ingress to zero the outer ECN field in any of the following cases:

- * if it is known that the tunnel egress does not support any of the RFCs that define propagation of the ECN field (RFC 6040, RFC 4301 or the full functionality mode of RFC 3168)
- * or if the behaviour of the egress is not known or an egress with unknown behaviour might be dynamically paired with the ingress.
- * or if an IP header might be encapsulated within a non-IP header that the tunnel ingress is encapsulating, but the ingress does not inspect within the encapsulation.

For the avoidance of doubt, the above only concerns the outer IP header. The ingress MUST NOT alter the ECN field of the arriving IP header that will become the inner IP header.

In order that the network operator can comply with the above safety rules, even if an implementation of a tunnel ingress does not claim to support RFC 6040, RFC 4301 or the full functionality mode of RFC 3168:

- * it MUST NOT treat the former ToS octet (IPv4) or the former Traffic Class octet (IPv6) as a single 8-bit field, as the resulting linkage of ECN and Diffserv field propagation between inner and outer is not consistent with the definition of the 6-bit Diffserv field in [RFC2474] and [RFC3260];
- * it SHOULD be able to be configured to zero the ECN field of the outer header.

"

For instance, if a tunnel ingress with no ECN-specific logic had a configuration capability to refer to the last 2 bits of the old ToS Byte of the outer (e.g. with a 0x3 mask) and set them to zero, while also being able to allow the DSCP to be re-mapped independently, that would be sufficient to satisfy both the above implementation requirements.

There might be concern that the above "MUST NOT" makes compliant implementations non-compliant at a stroke. However, by definition it solely applies to equipment that provides Diffserv configuration. Any such Diffserv equipment that is configuring treatment of the former ToS octet (IPv4) or the former Traffic Class octet (IPv6) as a single 8-bit field must have always been non-compliant with the definition of the 6-bit Diffserv field in [RFC2474] and [RFC3260]. If a tunnel ingress does not have any ECN logic, copying the ECN field as a side-effect of copying the DSCP is a seriously unsafe bug

that risks breaking the feedback loops that regulate load on a tunnel.

Zeroing the outer ECN field of all packets in all circumstances would be safe, but it would not be sufficient to claim compliance with RFC 6040 because it would not meet the aim of introducing ECN support to tunnels (see Section 4.3 of [RFC6040]).

5. ECN Propagation and Fragmentation/Reassembly

The following requirements update RFC6040, which omitted handling of the ECN field during fragmentation or reassembly. These changes might alter how many ECN-marked packets are propagated by a tunnel that fragments packets, but this would not raise any backward compatibility issues:

If a tunnel ingress fragments a packet, it MUST set the outer ECN field of all the fragments to the same value as it would have set if it had not fragmented the packet.

Section 5.3 of [RFC3168] specifies ECN requirements for reassembly of sets of outer fragments [I-D.ietf-intarea-tunnels] into packets. The following two additional requirements apply at a tunnel egress:

- o During reassembly of outer fragments [I-D.ietf-intarea-tunnels], if the ECN fields of the outer headers being reassembled into a single packet consist of a mixture of Not-ECT and other ECN codepoints, the packet MUST be discarded.
- o If there is mix of ECT(0) and ECT(1) fragments, then the reassembled packet MUST be set to either ECT(0) or ECT(1). In this case, reassembly SHOULD take into account that the RFC series has so far ensured that ECT(0) and ECT(1) can either be considered equivalent, or they can provide 2 levels of congestion severity, where the ranking of severity from highest to lowest is CE, ECT(1), ECT(0) [RFC6040].

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

There follows a list of specifications of encapsulations with tightly coupled shim header(s), in rough chronological order. The list is confined to standards track or widely deployed protocols. The list is not necessarily exhaustive so, for the avoidance of doubt, the scope of RFC 6040 is defined in Section 3 and is not limited to this list.

- o PPTP (Point-to-Point Tunneling Protocol) [RFC2637];

- o L2TP (Layer 2 Tunnelling Protocol), specifically L2TPv2 [RFC2661] and L2TPv3 [RFC3931], which not only includes all the L2-specific specializations of L2TP, but also derivatives such as the Keyed IPv6 Tunnel [RFC8159];
- o GRE (Generic Routing Encapsulation) [RFC2784] and NVGRE (Network Virtualization using GRE) [RFC7637];
- o GTP (GPRS Tunnelling Protocol), specifically GTPv1 [GTPv1], GTP v1 User Plane [GTPv1-U], GTP v2 Control Plane [GTPv2-C];
- o Teredo [RFC4380];
- o CAPWAP (Control And Provisioning of Wireless Access Points) [RFC5415];
- o LISP (Locator/Identifier Separation Protocol) [RFC6830];
- o AMT (Automatic Multicast Tunneling) [RFC7450];
- o VXLAN (Virtual eXtensible Local Area Network) [RFC7348] and VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe];
- o The Network Service Header (NSH [RFC8300]) for Service Function Chaining (SFC);
- o Geneve [RFC8926];
- o GUE (Generic UDP Encapsulation) [I-D.ietf-intarea-gue];
- o Direct tunnelling of an IP packet within a UDP/IP datagram (see Section 3.1.11 of [RFC8085]);
- o TCP Encapsulation of IKE and IPsec Packets (see Section 12.5 of [RFC8229]).

Some of the listed protocols enable encapsulation of a variety of network layer protocols as inner and/or outer. This specification applies in the cases where there is an inner and outer IP header as described in Section 3. Otherwise [I-D.ietf-tsvwg-ecn-encap-guidelines] gives guidance on how to design propagation of ECN into other protocols that might encapsulate IP.

Where protocols in the above list need to be updated to specify ECN propagation and they are under IETF change control, update text is given in the following subsections. For those not under IETF control, it is RECOMMENDED that implementations of encapsulation and decapsulation comply with RFC 6040. It is also RECOMMENDED that

their specifications are updated to add a requirement to comply with RFC 6040 (as updated by the present document).

PPTP is not under the change control of the IETF, but it has been documented in an informational RFC [RFC2637]. However, there is no need for the present specification to update PPTP because L2TP has been developed as a standardized replacement.

NVGRE is not under the change control of the IETF, but it has been documented in an informational RFC [RFC7637]. NVGRE is a specific use-case of GRE (it re-purposes the key field from the initial specification of GRE [RFC1701] as a Virtual Subnet ID). Therefore the text that updates GRE in Section 6.1.2 below is also intended to update NVGRE.

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.

The specification of CAPWAP already specifies RFC 3168 ECN propagation and ECN capability negotiation. Without modification the CAPWAP specification already interworks with the backward compatible updates to RFC 3168 in RFC 6040.

LISP made the ECN propagation procedures in RFC 3168 mandatory from the start. RFC 3168 has since been updated by RFC 6040, but the changes are backwards compatible so there is still no need for LISP tunnel endpoints to negotiate their ECN capabilities.

VXLAN is not under the change control of the IETF but it has been documented in an informational RFC. In contrast, VXLAN-GPE (Generic Protocol Extension) is being documented under IETF change control. It is RECOMMENDED that VXLAN and VXLAN-GPE implementations comply with RFC 6040 when the VXLAN header is inserted between (or removed from between) IP headers. The authors of any future update to these specifications are encouraged to add a requirement to comply with RFC 6040 as updated by the present specification.

The Network Service Header (NSH [RFC8300]) has been defined as a shim-based encapsulation to identify the Service Function Path (SFP) in the Service Function Chaining (SFC) architecture [RFC7665]. A proposal has been made for the processing of ECN when handling transport encapsulation [I-D.ietf-sfc-nsh-ecn-support].

The specifications of Geneve and GUE already refer to RFC 6040 for ECN encapsulation.

Section 3.1.11 of RFC 8085 already explains that a tunnel that encapsulates an IP header within a UDP/IP datagram needs to follow RFC 6040 when propagating the ECN field between inner and outer IP headers. The requirements in Section 4 update RFC 6040, and hence implicitly update the UDP usage guidelines in RFC 8085 to add the important but previously unstated requirement that, if the UDP tunnel egress does not, or might not, support ECN propagation, a UDP tunnel ingress has to clear the outer IP ECN field to 0b00, e.g. by configuration.

Section 12.5 of TCP Encapsulation of IKE and IPsec Packets [RFC8229] already recommends the compatibility mode of RFC 6040 in this case, because there is not a one-to-one mapping between inner and outer packets.

6.1. Specific Updates to Protocols under IETF Change Control

6.1.1. L2TP (v2 and v3) ECN Extension

The L2TP terminology used here is defined in [RFC2661] and [RFC3931].

L2TPv3 [RFC3931] is used as a shim header between any packet-switched network (PSN) header (e.g. IPv4, IPv6, MPLS) and many types of layer 2 (L2) header. The L2TPv3 shim header encapsulates an L2-specific sub-layer then an L2 header that is likely to contain an inner IP header (v4 or v6). Then this whole stack of headers can be encapsulated optionally within an outer UDP header then an outer PSN header that is typically IP (v4 or v6).

L2TPv2 is used as a shim header between any PSN header and a PPP header, which is in turn likely to encapsulate an IP header.

Even though these shims are rather fat (particularly in the case of L2TPv3), they still fit the definition of a tightly coupled shim header over an encapsulating header (Section 3.1), because all the headers encapsulating the L2 header are added (or removed) together. L2TPv2 and L2TPv3 are therefore within the scope of RFC 6040, as updated by Section 3 above.

L2TP maintainers are RECOMMENDED to implement the ECN extension to L2TPv2 and L2TPv3 defined in Section 6.1.1.2 below, in order to provide the benefits of ECN [RFC8087], whenever a node within an L2TP tunnel becomes the bottleneck for an end-to-end traffic flow.

6.1.1.1. Safe Configuration of a 'Non-ECN' Ingress LCCE

The following text is appended to both Section 5.3 of [RFC2661] and Section 4.5 of [RFC3931] as an update to the base L2TPv2 and L2TPv3 specifications:

The operator of an LCCE that does not support the ECN Extension in Section 6.1.1.2 of RFCXXXX MUST follow the configuration requirements in Section 4 of RFCXXXX to ensure it clears the outer IP ECN field to 0b00 when the outer PSN header is IP (v4 or v6). {RFCXXXX refers to the present document so it will need to be inserted by the RFC Editor}

In particular, for an LCCE implementation that does not support the ECN Extension, this means that configuration of how it propagates the ECN field between inner and outer IP headers MUST be independent of any configuration of the Diffserv extension of L2TP [RFC3308].

6.1.1.2. ECN Extension for L2TP (v2 or v3)

When the outer PSN header and the payload inside the L2 header are both IP (v4 or v6), to comply with RFC 6040, an LCCE will follow the rules for propagation of the ECN field at ingress and egress in Section 4 of RFC 6040 [RFC6040].

Before encapsulating any data packets, RFC 6040 requires an ingress LCCE to check that the egress LCCE supports ECN propagation as defined in RFC 6040 or one of its compatible predecessors ([RFC4301] or the full functionality mode of [RFC3168]). If the egress supports ECN propagation, the ingress LCCE can use the normal mode of encapsulation (copying the ECN field from inner to outer). Otherwise, the ingress LCCE has to use compatibility mode [RFC6040] (clearing the outer IP ECN field to 0b00).

An LCCE can determine the remote LCCE's support for ECN either statically (by configuration) or by dynamic discovery during setup of each control connection between the LCCEs, using the Capability AVP defined in Section 6.1.1.2.1 below.

Where the outer PSN header is some protocol other than IP that supports ECN, the appropriate ECN propagation specification will need to be followed, e.g. "Explicit Congestion Marking in MPLS" [RFC5129]. Where no specification exists for ECN propagation by a particular PSN, [I-D.ietf-tsvwg-ecn-encap-guidelines] gives general guidance on how to design ECN propagation into a protocol that encapsulates IP.

6.1.1.2.1. LCCE Capability AVP for ECN Capability Negotiation

The LCCE Capability Attribute-Value Pair (AVP) defined here has Attribute Type ZZ. The Attribute Value field for this AVP is a bit-mask with the following 16-bit format:

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|X X X X X X X X X X X X X X X E|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

Figure 1: Value Field for the LCCE Capability Attribute

This AVP MAY be present in the following message types: SCCRQ and SCCRP (Start-Control-Connection-Request and Start-Control-Connection-Reply). This AVP MAY be hidden (the H-bit set to 0 or 1) and is optional (M-bit not set). The length (before hiding) of this AVP MUST be 8 octets. The Vendor ID is the IETF Vendor ID of 0.

Bit 15 of the Value field of the LCCE Capability AVP is defined as the ECN Capability flag (E). When the ECN Capability flag is set to 1, it indicates that the sender supports ECN propagation. When the ECN Capability flag is cleared to zero, or when no LCCE Capability AVP is present, it indicates that the sender does not support ECN propagation. All the other bits are reserved. They MUST be cleared to zero when sent and ignored when received or forwarded.

An LCCE initiating a control connection will send a Start-Control-Connection-Request (SCCRQ) containing an LCCE Capability AVP with the ECN Capability flag set to 1. If the tunnel terminator supports ECN, it will return a Start-Control-Connection-Reply (SCCRP) that also includes an LCCE Capability AVP with the ECN Capability flag set to 1. Then, for any sessions created by that control connection, both ends of the tunnel can use the normal mode of RFC 6040, i.e. it can copy the IP ECN field from inner to outer when encapsulating data packets.

If, on the other hand, the tunnel terminator does not support ECN it will ignore the ECN flag in the LCCE Capability AVP and send an SCCRP to the tunnel initiator without a Capability AVP (or with a Capability AVP but with the ECN Capability flag cleared to zero). The tunnel initiator interprets the absence of the ECN Capability flag in the SCCRP as an indication that the tunnel terminator is incapable of supporting ECN. When encapsulating data packets for any sessions created by that control connection, the tunnel initiator will then use the compatibility mode of RFC 6040 to clear the ECN field of the outer IP header to 0b00.

If the tunnel terminator does not support this ECN extension, the network operator is still expected to configure it to comply with the safety provisions set out in Section 6.1.1.1 above, when it acts as an ingress LCCE.

6.1.2. GRE

The GRE terminology used here is defined in [RFC2784]. GRE is often used as a tightly coupled shim header between IP headers. Sometimes the GRE shim header encapsulates an L2 header, which might in turn encapsulate an IP header. Therefore GRE is within the scope of RFC 6040 as updated by Section 3 above.

GRE tunnel endpoint maintainers are RECOMMENDED to support [RFC6040] as updated by the present specification, in order to provide the benefits of ECN [RFC8087] whenever a node within a GRE tunnel becomes the bottleneck for an end-to-end IP traffic flow tunnelled over GRE using IP as the delivery protocol (outer header).

GRE itself does not support dynamic set-up and configuration of tunnels. However, control plane protocols such as Mobile IPv4 (MIP4) [RFC5944], Mobile IPv6 (MIP6) [RFC6275], Proxy Mobile IP (PMIP) [RFC5845] and IKEv2 [RFC7296] are sometimes used to set up GRE tunnels dynamically.

When these control protocols set up IP-in-IP or IPSec tunnels, it is likely that they propagate the ECN field as defined in RFC 6040 or one of its compatible predecessors (RFC 4301 or the full functionality mode of RFC 3168). However, if they use a GRE encapsulation, this presumption is less sound.

Therefore, If the outer delivery protocol is IP (v4 or v6) the operator is obliged to follow the safe configuration requirements in Section 4 above. Section 6.1.2.1 below updates the base GRE specification with this requirement, to emphasize its importance.

Where the delivery protocol is some protocol other than IP that supports ECN, the appropriate ECN propagation specification will need to be followed, e.g Explicit Congestion Marking in MPLS [RFC5129]. Where no specification exists for ECN propagation by a particular PSN, [I-D.ietf-tsvwg-ecn-encap-guidelines] gives more general guidance on how to propagate ECN to and from protocols that encapsulate IP.

6.1.2.1. Safe Configuration of a 'Non-ECN' GRE Ingress

The following text is appended to Section 3 of [RFC2784] as an update to the base GRE specification:

The operator of a GRE tunnel ingress MUST follow the configuration requirements in Section 4 of RFCXXXX when the outer delivery protocol is IP (v4 or v6). {RFCXXXX refers to the present document so it will need to be inserted by the RFC Editor}

6.1.3. Teredo

Teredo [RFC4380] provides a way to tunnel IPv6 over an IPv4 network, with a UDP-based shim header between the two.

For Teredo tunnel endpoints to provide the benefits of ECN, the Teredo specification would have to be updated to include negotiation of the ECN capability between Teredo tunnel endpoints. Otherwise it would be unsafe for a Teredo tunnel ingress to copy the ECN field to the IPv6 outer.

It is believed that current implementations do not support propagation of ECN, but that they do safely zero the ECN field in the outer IPv6 header. However the specification does not mention anything about this.

To make existing Teredo deployments safe, it would be possible to add ECN capability negotiation to those that are subject to remote OS update. However, for those implementations not subject to remote OS update, it will not be feasible to require them to be configured correctly, because Teredo tunnel endpoints are generally deployed on hosts.

Therefore, until ECN support is added to the specification of Teredo, the only feasible further safety precaution available here is to update the specification of Teredo implementations with the following text, as a new section 5.1.3:

"5.1.3 Safe 'Non-ECN' Teredo Encapsulation

A Teredo tunnel ingress implementation that does not support ECN propagation as defined in RFC 6040 or one of its compatible predecessors (RFC 4301 or the full functionality mode of RFC 3168) MUST zero the ECN field in the outer IPv6 header."

6.1.4. AMT

Automatic Multicast Tunneling (AMT [RFC7450]) is a tightly coupled shim header that encapsulates an IP packet and is itself encapsulated within a UDP/IP datagram. Therefore AMT is within the scope of RFC 6040 as updated by Section 3 above.

AMT tunnel endpoint maintainers are RECOMMENDED to support [RFC6040] as updated by the present specification, in order to provide the benefits of ECN [RFC8087] whenever a node within an AMT tunnel becomes the bottleneck for an IP traffic flow tunnelled over AMT.

To comply with RFC 6040, an AMT relay and gateway will follow the rules for propagation of the ECN field at ingress and egress respectively, as described in Section 4 of RFC 6040 [RFC6040].

Before encapsulating any data packets, RFC 6040 requires an ingress AMT relay to check that the egress AMT gateway supports ECN propagation as defined in RFC 6040 or one of its compatible predecessors (RFC 4301 or the full functionality mode of RFC 3168). If the egress gateway supports ECN, the ingress relay can use the normal mode of encapsulation (copying the IP ECN field from inner to outer). Otherwise, the ingress relay has to use compatibility mode, which means it has to clear the outer ECN field to zero [RFC6040].

An AMT tunnel is created dynamically (not manually), so the relay will need to determine the remote gateway's support for ECN using the ECN capability declaration defined in Section 6.1.4.2 below.

6.1.4.1. Safe Configuration of a 'Non-ECN' Ingress AMT Relay

The following text is appended to Section 4.2.2 of [RFC7450] as an update to the AMT specification:

The operator of an AMT relay that does not support RFC 6040 or one of its compatible predecessors (RFC 4301 or the full functionality mode of RFC 3168) MUST follow the configuration requirements in Section 4 of RFCXXXX to ensure it clears the outer IP ECN field to zero. {RFCXXXX refers to the present document so it will need to be inserted by the RFC Editor}

6.1.4.2. ECN Capability Declaration of an AMT Gateway

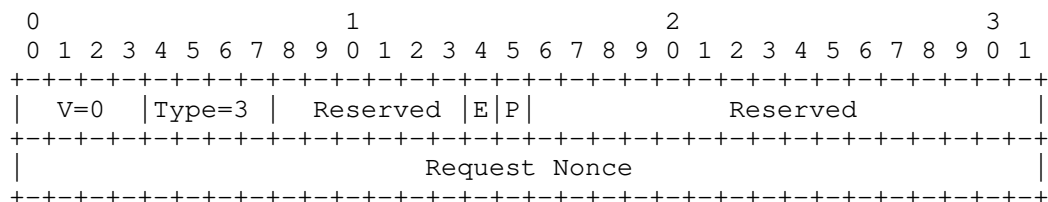


Figure 2: Updated AMT Request Message Format

Bit 14 of the AMT Request Message counting from 0 (or bit 7 of the Reserved field counting from 1) is defined here as the AMT Gateway ECN Capability flag (E), as shown in Figure 2. The definitions of all other fields in the AMT Request Message are unchanged from RFC 7450.

When the E flag is set to 1, it indicates that the sender of the message supports RFC 6040 ECN propagation. When it is cleared to zero, it indicates the sender of the message does not support RFC 6040 ECN propagation. An AMT gateway "that supports RFC 6040 ECN propagation" means one that propagates the ECN field to the forwarded data packet based on the combination of arriving inner and outer ECN fields, as defined in Section 4 of RFC 6040.

The other bits of the Reserved field remain reserved. They will continue to be cleared to zero when sent and ignored when either received or forwarded, as specified in Section 5.1.3.3. of RFC 7450.

An AMT gateway that does not support RFC 6040 MUST NOT set the E flag of its Request Message to 1.

An AMT gateway that supports RFC 6040 ECN propagation MUST set the E flag of its Relay Discovery Message to 1.

The action of the corresponding AMT relay that receives a Request message with the E flag set to 1 depends on whether the relay itself supports RFC 6040 ECN propagation:

- o If the relay supports RFC 6040 ECN propagation, it will store the ECN capability of the gateway along with its address. Then whenever it tunnels datagrams towards this gateway, it MUST use the normal mode of RFC 6040 to propagate the ECN field when encapsulating datagrams (i.e. it copies the IP ECN field from inner to outer).

- o If the discovered AMT relay does not support RFC 6040 ECN propagation, it will ignore the E flag in the Reserved field, as per section 5.1.3.3. of RFC 7450.

If the AMT relay does not support RFC 6040 ECN propagation, the network operator is still expected to configure it to comply with the safety provisions set out in Section 6.1.4.1 above.

7. IANA Considerations

IANA is requested to assign the following L2TP Control Message Attribute Value Pair:

Attribute Type	Description	Reference
ZZ	ECN Capability	RFCXXXX

[TO BE REMOVED: This registration should take place at the following location: <https://www.iana.org/assignments/l2tp-parameters/l2tp-parameters.xhtml>]

8. Security Considerations

The Security Considerations in [RFC6040] and [I-D.ietf-tsvwg-ecn-encap-guidelines] apply equally to the scope defined for the present specification.

9. 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.

10. Acknowledgements

Thanks to Ing-jyh (Inton) Tsang for initial discussions on the need for ECN propagation in L2TP and its applicability. Thanks also to Carlos Pignataro, Tom Herbert, Ignacio Goyret, Alia Atlas, Praveen Balasubramanian, Joe Touch, Mohamed Boucadair, David Black, Jake Holland and Sri Gundavelli for helpful advice and comments. "A Comparison of IPv6-over-IPv4 Tunnel Mechanisms" [RFC7059] helped to identify a number of tunnelling protocols to include within the scope of this document.

Bob Briscoe was part-funded by the Research Council of Norway through the TimeIn project. The views expressed here are solely those of the authors.

11. References

11.1. Normative References

- [I-D.ietf-tsvwg-ecn-encap-guidelines]
Briscoe, B. and J. Kaippallimalil, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP", draft-ietf-tsvwg-ecn-encap-guidelines-15 (work in progress), March 2021.
- [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>>.
- [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, <<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, DOI 10.17487/RFC2661, August 1999, <<https://www.rfc-editor.org/info/rfc2661>>.
- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <<https://www.rfc-editor.org/info/rfc2784>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3931] Lau, J., Ed., Townsley, M., Ed., and I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, DOI 10.17487/RFC3931, March 2005, <<https://www.rfc-editor.org/info/rfc3931>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.

- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <<https://www.rfc-editor.org/info/rfc4380>>.
- [RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <<https://www.rfc-editor.org/info/rfc5129>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.

11.2. Informative 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-intarea-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-09 (work in progress), October 2019.
- [I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.
- [I-D.ietf-nvo3-vxlan-gpe]
(Editor), F. M., (editor), L. K., and U. E. (editor), "Generic Protocol Extension for VXLAN (VXLAN-GPE)", draft-ietf-nvo3-vxlan-gpe-11 (work in progress), March 2021.
- [I-D.ietf-sfc-nsh-ecn-support]
Eastlake, D. E., Briscoe, B., Li, Y., Malis, A. G., and X. Wei, "Explicit Congestion Notification (ECN) and Congestion Feedback Using the Network Service Header (NSH) and IPFIX", draft-ietf-sfc-nsh-ecn-support-05 (work in progress), April 2021.

- [RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, DOI 10.17487/RFC1701, October 1994, <<https://www.rfc-editor.org/info/rfc1701>>.
- [RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, DOI 10.17487/RFC2637, July 1999, <<https://www.rfc-editor.org/info/rfc2637>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<https://www.rfc-editor.org/info/rfc2983>>.
- [RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, DOI 10.17487/RFC3260, April 2002, <<https://www.rfc-editor.org/info/rfc3260>>.
- [RFC3308] Calhoun, P., Luo, W., McPherson, D., and K. Peirce, "Layer Two Tunneling Protocol (L2TP) Differentiated Services Extension", RFC 3308, DOI 10.17487/RFC3308, November 2002, <<https://www.rfc-editor.org/info/rfc3308>>.
- [RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <<https://www.rfc-editor.org/info/rfc5415>>.
- [RFC5845] Muhanna, A., Khalil, M., Gundavelli, S., and K. Leung, "Generic Routing Encapsulation (GRE) Key Option for Proxy Mobile IPv6", RFC 5845, DOI 10.17487/RFC5845, June 2010, <<https://www.rfc-editor.org/info/rfc5845>>.
- [RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised", RFC 5944, DOI 10.17487/RFC5944, November 2010, <<https://www.rfc-editor.org/info/rfc5944>>.
- [RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <<https://www.rfc-editor.org/info/rfc6275>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.

- [RFC7059] Steffann, S., van Beijnum, I., and R. van Rein, "A Comparison of IPv6-over-IPv4 Tunnel Mechanisms", RFC 7059, DOI 10.17487/RFC7059, November 2013, <<https://www.rfc-editor.org/info/rfc7059>>.
- [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, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC7450] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, DOI 10.17487/RFC7450, February 2015, <<https://www.rfc-editor.org/info/rfc7450>>.
- [RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <<https://www.rfc-editor.org/info/rfc7637>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <<https://www.rfc-editor.org/info/rfc8087>>.
- [RFC8159] Konstantynowicz, M., Ed., Heron, G., Ed., Schatzmayr, R., and W. Henderickx, "Keyed IPv6 Tunnel", RFC 8159, DOI 10.17487/RFC8159, May 2017, <<https://www.rfc-editor.org/info/rfc8159>>.
- [RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <<https://www.rfc-editor.org/info/rfc8229>>.

- [RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed.,
"Network Service Header (NSH)", RFC 8300,
DOI 10.17487/RFC8300, January 2018,
<<https://www.rfc-editor.org/info/rfc8300>>.
- [RFC8926] Gross, J., Ed., Ganga, I., Ed., and T. Sridhar, Ed.,
"Geneve: Generic Network Virtualization Encapsulation",
RFC 8926, DOI 10.17487/RFC8926, November 2020,
<<https://www.rfc-editor.org/info/rfc8926>>.

Author's Address

Bob Briscoe
Independent
UK

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

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.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Overview	3
1.2. Conventions	5
2. User Message Interleaving	5
2.1. The I-DATA Chunk Supporting User Message Interleaving . .	6
2.2. Procedures	8
2.2.1. Negotiation	9
2.2.2. Sender Side Considerations	9
2.2.3. Receiver Side Considerations	10
2.3. Interaction with other SCTP Extensions	10
2.3.1. SCTP Partial Reliability Extension	10
2.3.2. SCTP Stream Reconfiguration Extension	12
3. Stream Schedulers	12
3.1. First Come First Served Scheduler (SCTP_SS_FCFS)	13
3.2. Round Robin Scheduler (SCTP_SS_RR)	13
3.3. Round Robin Scheduler per Packet (SCTP_SS_RR_PKT)	13
3.4. Priority Based Scheduler (SCTP_SS_PRIO)	13
3.5. Fair Capacity Scheduler (SCTP_SS_FC)	14
3.6. Weighted Fair Queueing Scheduler (SCTP_SS_WFQ)	14
4. Socket API Considerations	14
4.1. Exposure of the Stream Sequence Number (SSN)	14
4.2. SCTP_ASSOC_CHANGE Notification	15
4.3. Socket Options	15
4.3.1. Enable or Disable the Support of User Message Interleaving (SCTP_INTERLEAVING_SUPPORTED)	15
4.3.2. Get or Set the Stream Scheduler (SCTP_STREAM_SCHEDULER)	16
4.3.3. Get or Set the Stream Scheduler Parameter (SCTP_STREAM_SCHEDULER_VALUE)	17
4.4. Explicit EOR Marking	18
5. IANA Considerations	18

5.1. I-DATA Chunk	18
5.2. I-FORWARD-TSN Chunk	19
6. Security Considerations	19
7. Acknowledgments	20
8. References	20
8.1. Normative References	20
8.2. Informative References	21
Authors' Addresses	21

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.

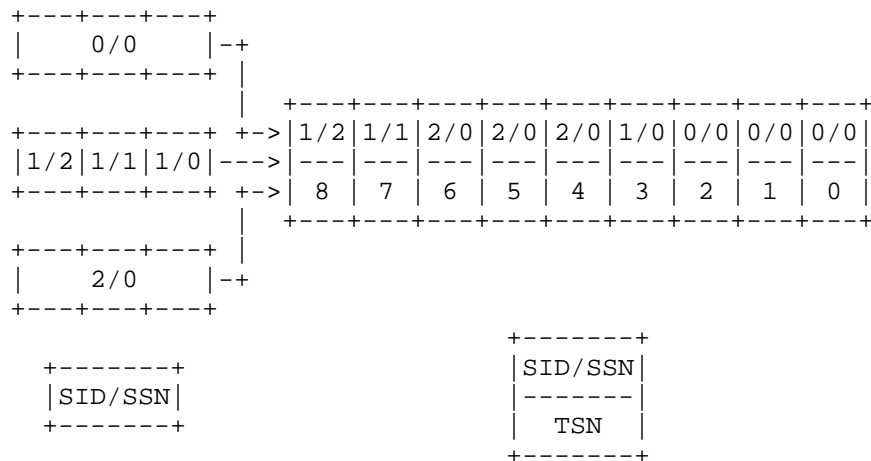


Figure 1: Round Robin Scheduler without User Message Interleaving

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.

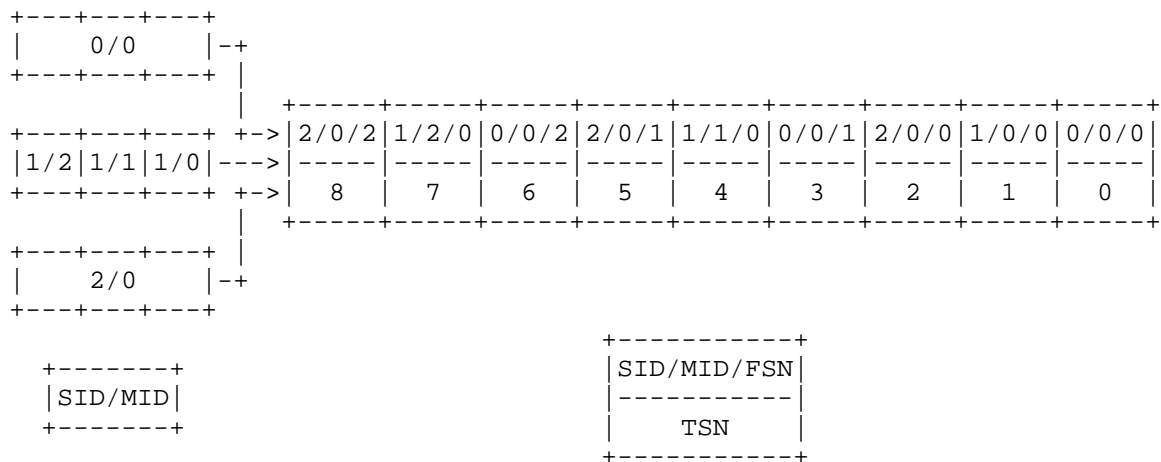


Figure 2: Round Robin Scheduler with User Message Interleaving

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.

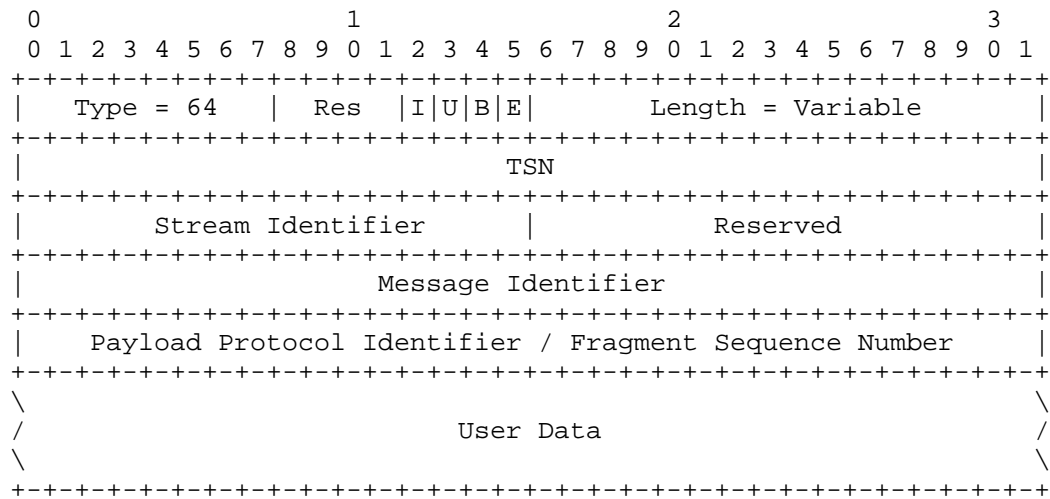


Figure 3: I-DATA chunk format

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)mmmediate 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 one 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.

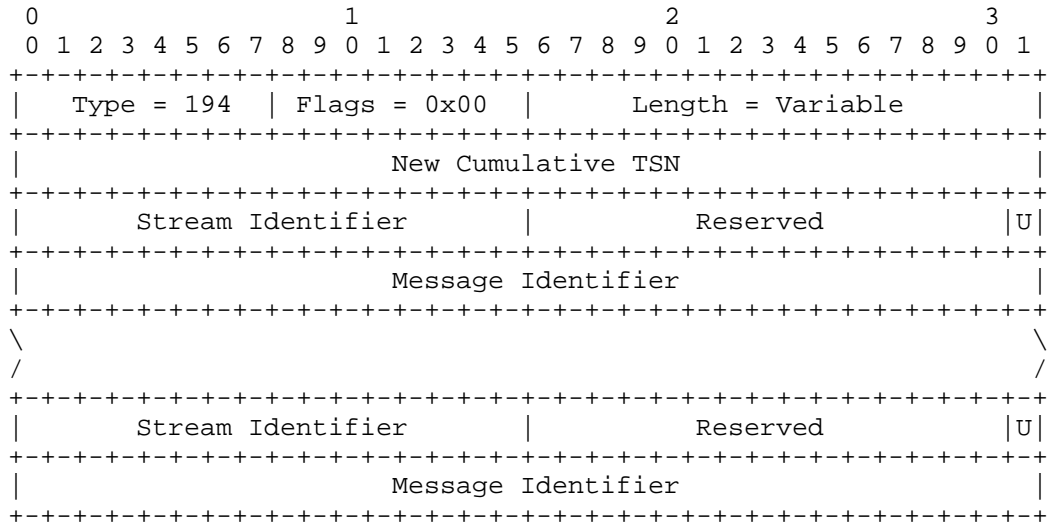


Figure 4: I-FORWARD-TSN chunk format

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

Internet Engineering Task Force
INTERNET-DRAFT
Intended Status: Informational
Expires: November 7, 2019

X. Wei
Y. Li
Huawei Technologies
S. Boutros
VMware
L. Geng
China Mobile
May 6, 2019

Tunnel Congestion Feedback
draft-ietf-tsvwg-tunnel-congestion-feedback-07

Abstract

This document describes a method to measure congestion on a tunnel segment based on recommendations from RFC 6040, "Tunneling of Explicit Congestion Notification", and to use IPFIX to communicate the congestion measurements from the tunnel's egress to a controller which can respond by modifying the traffic control policies at the tunnel's ingress.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions And Terminologies	3
3. Congestion Information Feedback Models	4
4. Congestion Level Measurement	5
5. Congestion Information Delivery	7
5.1 IPFIX Extensions	8
5.1.1 tunnelEcnCeCeByteTotalCount	8
5.1.2 tunnelEcnEct0NectBytetTotalCount	8
5.1.3 tunnelEcnEct1NectByteTotalCount	9
5.1.4 tunnelEcnCeNectByteTotalCount	9
5.1.5 tunnelEcnCeEct0ByteTotalCount	9
5.1.6 tunnelEcnCeEct1ByteTotalCount	10
5.1.7 tunnelEcnEct0Ect0ByteTotalCount	10
5.1.8 tunnelEcnEct1Ect1PacketTotalCount	10
5.1.9 tunnelEcnCEMarkedRatio	11
6. Congestion Management	11
6.1 Example	11
7. Security Considerations	14
8. IANA Considerations	15
9. References	17
9.1 Normative References	17
9.2 Informative References	18
10. Acknowledgements	18
Authors' Addresses	18

1. Introduction

In IP networks, persistent congestion[RFC2914] lowers transport throughput, leading to waste of network resource. Appropriate congestion control mechanisms are therefore critical to prevent the network from falling into the persistent congestion state. Currently, transport protocols such as TCP[RFC793], SCTP[RFC4960], DCCP[RFC4340], have their built-in congestion control mechanisms, and even for certain single transport protocol like TCP there can be a couple of different congestion control mechanisms to choose from. All these congestion control mechanisms are implemented on host side, and there are reasons that only host side congestion control is not sufficient for the whole network to keep away from persistent congestion. For example, (1) some protocol's congestion control scheme may have internal design flaws; (2) improper software implementation of protocol; (3) some transport protocols, e.g. RTP[RFC3550] do not even provide congestion control at all; (4) a heavy load from a much larger than expected number of responsive flows could also lead to persistent congestion.

Tunnels are widely deployed in various networks including public Internet, data center network, and enterprise network etc. A tunnel consists of ingress, egress and a set of intermediate routers. For the tunnel scenario, a tunnel-based mechanism is introduced for network traffic control to keep the network from persistent congestion. Here, tunnel ingress will implement congestion management function to control the traffic entering the tunnel.

This document provides a mechanism of feeding back inner tunnel congestion level to the ingress. Using this mechanism the egress can feed the tunnel congestion level information it collects back to the ingress. After receiving this information the ingress will be able to perform congestion management according to network management policy.

The following subjects are out of scope of current document: it gives no advice on how to select which tunnel endpoints should be used in order to manage traffic over a network criss-crossed by multiple tunnels; if a congested node is part of multiple tunnels, and it causes congestion feedback to multiple traffic management functions at the ingresses of all the tunnels, the draft gives no advice on how all the traffic management functions should respond.

2. Conventions And Terminologies

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

DP: Decision Point, an logical entity that makes congestion management decision based on the received congestion feedback information.

EP: Enforcement Point, an logical entity that implements congestion management action according to the decision made by Decision Point.

ECT: ECN-Capable Transport code point defined in RFC3168.

3. Congestion Information Feedback Models

The feedback model mainly consists of tunnel egress and tunnel ingress. The tunnel egress composes of meter function and exporter function; tunnel ingress composes EP (Enforcement Point) function, collector function and DP (Decision Point) function.

The Meter function collects network congestion level information, and conveys the information to Exporter which feeds back the information to the collector function.

The feedback message contains CE-marked packet ratio, the traffic volumes of all kinds of ECN marking packets.

The collector collects congestion level information from exporter, after that congestion management Decision Point (DP) function will make congestion management decision based on the information from collector.

The Enforcement Point controls the traffic entering tunnel, and it implements traffic control decision of DP.

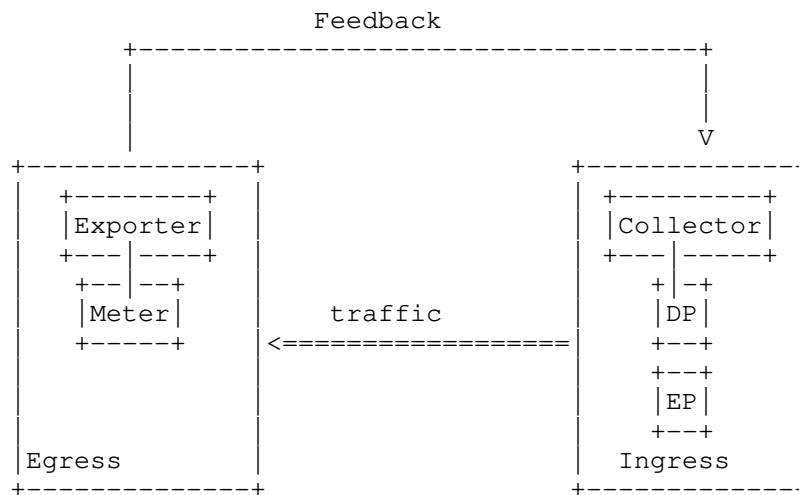


Figure 1: Feedback Model.

4. Congestion Level Measurement

The congestion level measurement is based on ECN (Explicit Congestion Notification) [RFC3168] and packet drop. The network congestion level could be indicated through the ratio of CE-marked packet and the volumes of packet drop, the relationship between these two kinds of indicator is complementary. If the congestion level in tunnel is not high enough, the packets would be marked as CE instead of being dropped, and then it is easy to calculate congestion level according to the ratio of CE-marked packets. If the congestion level is so high that ECT packet will be dropped, then the packet loss ratio could be calculated by comparing total packets entering ingress and total packets arriving at egress over the same span of packets, if packet loss is detected, it could be assumed that severe congestion has occurred in the tunnel.

Egress calculates CE-marked packet ratio by counting different kinds of ECN-marked packet, the CE-marked packet ratio will be used as an indication of tunnel load level. It's assumed that routers in the tunnel will not drop packets biased towards certain ECN codepoint, so calculating of CE-marked packet ratio is not affect by packet drop.

The calculation of volumes of packet drop is by comparing the traffic volumes between ingress and egress.

Faked ECN-capable transport (ECT) is used at ingress to defer

packet loss to egress. The basic idea of faked ECT is that, when encapsulating packets, ingress first marks tunnel outer header according to RFC6040, and then remarks outer header of Not-ECT packet as ECT, there will be three kinds of combination of outer header ECN field and inner header ECN field: CE|CE, ECT|N-ECT, ECT|ECT (in the form of outer ECN| inner ECN); when decapsulating packets at egress, RFC6040 defined decapsulation behavior is used, and according to RFC6040, the packets marked as CE|N-ECT will be dropped by egress. Faked-ECT is used to shift some drops to the egress in order to calculate CE-marked packet ratio more precisely by egress.

To calculate congestion level, for the same span of packets, the ratio of CE-marked packets will be calculated by egress, and the total bytes count of packets at ingress and egress will be compared to detect the traffic volume loss in tunnel.

The basic procedure of packets loss measurement is as follows:

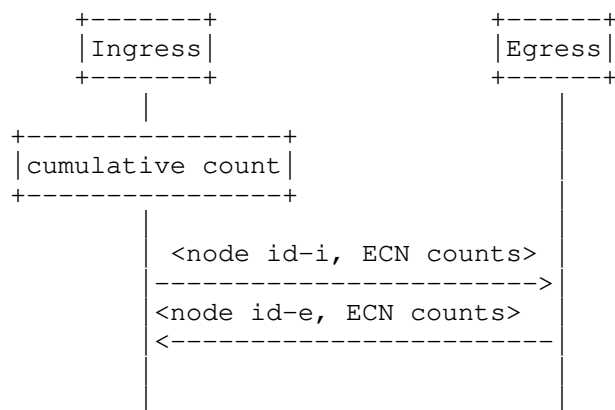


Figure 2: Procedure of Packet Loss Measurement

Ingress encapsulates packets and marks outer header according to faked ECT as described above. Ingress cumulatively counts packet bytes for three types of ECN combination (CE|CE, ECT|N-ECT, ECT|ECT) and then the ingress regularly sends cumulative bytes counts message of each type of ECN combination to the egress.

When each message arrives at egress, (1)egress calculates the ratio of CE-marked packet; (2)the egress cumulatively counts packet bytes coming from the ingress and adds its own bytes counts of each type of ECN combination (CE|CE, ECT|N-ECT, CE|N-ECT, CE|ECT, ECT|ECT) to the

message for ingress to calculate packet loss. Egress feeds back CE-marked packet ratio and bytes counts information to the ingress for evaluating congestion level in the tunnel.

The counting of bytes can be at the granularity of the all traffic from the ingress to the egress to learn about the overall congestion status of the path between the ingress and the egress. The counting can also be at the granularity of individual customer's traffic or a specific set of flows to learn about their congestion contribution.

5. Congestion Information Delivery

As described above, the tunnel ingress needs to convey a message containing cumulative bytes counts of packets of each type of ECN combination to tunnel egress, and the tunnel egress also needs to feed back the message of cumulative bytes counts of packets of each type of ECN combination and CE-marked packet ratio to the ingress. This section describes how the messages should be conveyed.

The message travels along the same path with network data traffic, referred as in-band signal. Because the message is transmitted in band, so the message packet may get lost in case of network congestion. To cope with the situation that the message packet gets lost, the bytes counts values are sent as cumulative counters. Then if a message is lost the next message will recover the missing information. Even though the missing information could be recovered, the message should be transmitted in a much higher priority than users' traffic flows.

IPFIX [RFC7011] is selected as a candidate information feedback protocol. IPFIX uses preferably SCTP as transport. SCTP allows partially reliable delivery [RFC3758], which ensures the feedback message will not be blocked in case of packet loss due to network congestion.

Ingress can do congestion management at different granularity which means both the overall aggregated inner tunnel congestion level and congestion level contributed by certain traffic(s) could be measured for different congestion management purpose. For example, if the ingress only wants to limit congestion volume caused by certain traffic(s), e.g. UDP-based traffic, then congestion volume for the traffic will be fed back; or if the ingress do overall congestion management, the aggregated congestion volume will be fed back.

When sending message from ingress to egress, the ingress acts as IPFIX exporter and egress acts as IPFIX collector; When feedback congestion level information from egress to ingress, then the egress acts as IPFIX exporter and ingress acts as IPFIX collector.

The combination of congestion level measurement and congestion information delivery procedure should be as following:

The ingress determines IPFIX template record to be used. The template record can be pre-configured or determined at runtime, the content of template record will be determined according to the granularity of congestion management, if the ingress wants to limit congestion volume contributed by specific traffic flow then the elements such as source IP address, destination IP address, flow id and CE-marked packet volume of the flow etc will be included in the template record.

Meter on ingress measures traffic volume according to template record chosen and then the measurement records are sent to egress in band.

Meter on egress measures congestion level information according to template record, the content of template record should be the same as template record of ingress.

Exporter of egress sends measurement record together with the measurement record of ingress back to the ingress.

5.1 IPFIX Extensions

This sub-section defines a list of new IPFIX Information Elements according to RFC7013 [RFC7013].

5.1.1 tunnelEcnCeCeByteTotalCount

Description: The total number of bytes of incoming packets with CE|CE ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD1

Statuses: current

Units: bytes

5.1.2 tunnelEcnEct0NectBytetTotalCount

Description: The total number of bytes of incoming packets with ECT(0)|N-ECT ECN marking combination at the Observation Point since

the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD2

Statuses: current

Units: bytes

5.1.3 tunnelEcnEct1NectByteTotalCount

Description: The total number of bytes of incoming packets with ECT(1)|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD3

Statuses: current

Units: bytes

5.1.4 tunnelEcnCeNectByteTotalCount

Description: The total number of bytes of incoming packets with CE|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD4

Statuses: current

Units: bytes

5.1.5 tunnelEcnCeEct0ByteTotalCount

Description: The total number of bytes of incoming packets with CE|ECT(0) ECN marking combination at the Observation Point since the

Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD5

Statues: current

Units: bytes

5.1.6 tunnelEcnCeEct1ByteTotalCount

Description: The total number of bytes of incoming packets with CE|ECT(1) ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD6

Statues: current

Units: bytes

5.1.7 tunnelEcnEct0Ect0ByteTotalCount

Description: The total number of bytes of incoming packets with ECT(0)|ECT(0) ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD7

Statues: current

Units: bytes

5.1.8 tunnelEcnEct1Ect1PacketTotalCount

Description: The total number of bytes of incoming packets with ECT(1)|ECT(1) ECN marking combination at the Observation Point since

the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD8

Statuses: current

Units: bytes

5.1.9 tunnelEcnCEMarkedRatio

Description: The ratio of CE-marked Packet at the Observation Point.

Abstract Data Type: float32

ElementId: TBD8

Statuses: current

6. Congestion Management

After tunnel ingress receives congestion level information, then congestion management actions could be taken based on the information, e.g. if the congestion level is higher than a predefined threshold, then action could be taken to reduce the congestion level.

The design of network side congestion management SHOULD take host side e2e congestion control mechanism into consideration, which means the congestion management needs to avoid the impacts on e2e congestion control. For instance, congestion management action must be delayed by more than a worst-case global RTT (e.g. 100ms), otherwise tunnel traffic management will not give normal e2e congestion control enough time to do its job, and the system could go unstable.

The detailed description of congestion management is out of scope of this document, as examples, congestion management such as circuit breaker [RFC8084] could be applied. Circuit breaker is an automatic mechanism to estimate congestion, and to terminate flow(s) when persistent congestion is detected to prevent network congestion collapse.

6.1 Example

This subsection provides an example of how the solution described in this document could work.

First of all, IPFIX template records are exchanged between ingress and egress to negotiate the format of data record, the example here is to measure the congestion level for the overall tunnel (caused by all the traffic in tunnel). After the negotiation is finished, ingress sends in-band message to egress, the message contains the number of each kind of ECN-marked packets (i.e. CE|CE, ECT|N-ECT and ECT|ECT) received until the sending of message.

After egress receives the message, the egress calculates CE-marked packet ratio and counts number of different kinds of ECN-marking packets received until receiving the message, then the egress sends a feedback message containing the counts together with the information in ingress's message to ingress.

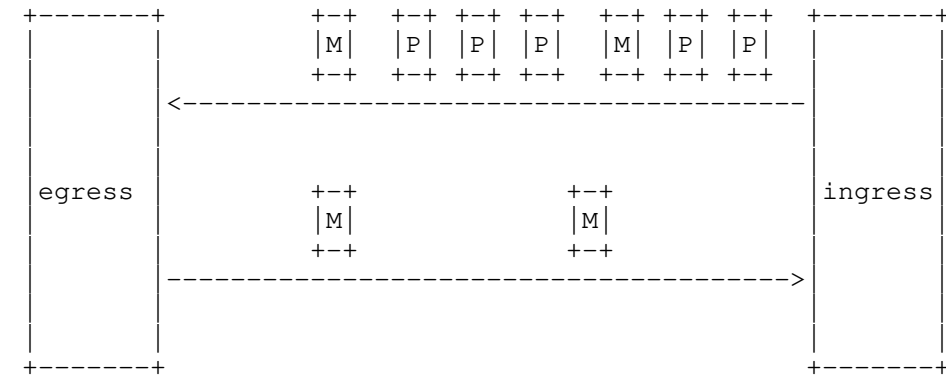
Figure 3 to Figure 6 below show the example procedure between ingress and egress.

Set ID=2	Length=40
Template ID=256	Field Count =8
tunnelEcnCeCeByteTotalCount	Field Length=8
tunnelEcnEctNectByteTotalCount	Field Length=8
tunnelEcnEctEctByteTotalCount	Field Length=8
tunnelEcnCeCeByteTotalCount	Field Length=8
tunnelEcnEctNectByteTotalCount	Field Length=8
tunnelEcnEctEctByteTotalCount	Field Length=8
tunnelEcnCeNectByteTotalCount	Field Length=8
tunnelEcnCeEctByteTotalCount	Field Length=8
tunnelEcnCEMarkedRatio	Field Length=4

Figure 3: Template Record Sent From Egress to Ingress

Set ID=2	Length=28
Template ID=257	Field Count =3
tunnelEcnCeCeByteTotalCount	Field Length=8
tunnelEcnEctNectByteTotalCount	Field Length=8
tunnelEcnEctEctByteTotalCount	Field Length=8

Figure 4: Template Record Sent From Ingress to Egress



+-+
 |M| : Message Packet
 +-+

+-+
 |P| : User Packet
 +-+

Figure 5 Traffic flow Between Ingress and Egress

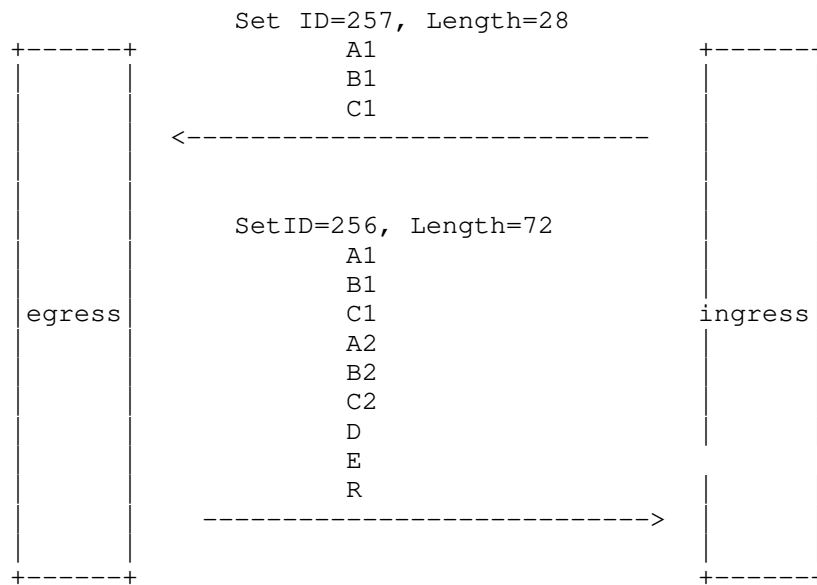


Figure 6: Message Between Ingress and Egress

The following provides an example of how tunnel congestion level could be calculated:

Congestion Level could be divided into two categories: (1) slight congestion (no packets dropped); (2) serious congestion (packet dropping happen).

For slight congestion, the congestion level is indicated as the ratio of CE-marked packet:

$ce_marked = R;$

For serious congestion, the congestion level is indicated as the number of volume loss:

$total_ingress = (A1 + B1 + C1)$

$total_egress = (A2 + B2 + C2 + D + E)$

$volume_loss = (total_ingress - total_egress)$

7. Security Considerations

This document describes the tunnel congestion calculation and feedback.

The tunnel endpoints are assumed to be deployed in the same administrative domain, so the ingress and egress will trust each other, the signaling traffic between ingress and egress will be protected utilizing security mechanism provided IPFIX (see section 11 in RFC7011).

From the consideration of privacy point of view, in case of fine grained congestion management, ingress is aware of the amount of traffic for specific application flows inside the tunnel which seems to be an invasion of privacy. But in any way, the ingress could The solution doesn't introduce more privacy problem.

8. IANA Considerations

This document defines a set of new IPFIX Information Elements (IE), which need to be registered at IANA IPFIX Information Element Registry.

ElementID: TBD1

Name: tunnelEcnCeCePacketTotalCount

Data Type: unsigned64

Data Type Semantics: totalCounter

Status: current

Description: The total number of bytes of incoming packets with CE|CE ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Units: octets

ElementID: TBD2

Name: tunnelEcnEct0NectPacketTotalCount

Data Type: unsigned64

Data Type Semantics: totalCounter

Status: current

Description: The total number of bytes of incoming packets with ECT(0)|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Units: octets

ElementID: TBD3

Name: tunnelEcnEct1NectPacketTotalCount

Data Type: unsigned64

Data Type Semantics: totalCounter

Status: current

Description: The total number of bytes of incoming packets with

ECT(1)|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD4
Name:tunnelEcnCeNectPacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with CE|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD5
Name:tunnelEcnCeEct0PacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with CE|ECT(0) ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD6
Name:tunnelEcnCeEct1PacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with CE|ECT(1) ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD7
Name:tunnelEcnEct0Ect0PacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description:The total number of bytes of incoming packets with ECT(0)|ECT(0) ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD8
Name:tunnelEcnEct1Ect1PacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter

Status: current

Description: The total number of bytes of incoming packets with ECT(1) | ECT(1)ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD9

Name: tunnelEcnCEMarkedRatio

Data Type: float32

Status: current

Description: The ratio of CE-marked Packet at the Observation Point.

[TO BE REMOVED: This registration should take place at the following location: <http://www.iana.org/assignments/ipfix/ipfix.xhtml#ipfix-information-elements>]

9. References

9.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.
- [CONEX] Matt Mathis, Bob Briscoe. "Congestion Exposure (ConEx) Concepts, Abstract Mechanism and Requirements", RFC7713, December 2015

9.2 Informative References

[RFC8084] G. Fairhurst. "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-01, April 02, 2015

10. Acknowledgements

Thanks Bob Briscoe for his insightful suggestions on the basic mechanisms of congestion information collection and many other useful comments. Thanks David Black for his useful technical suggestions. Also, thanks Lei Zhu, Lingli Deng, Anthony Chan, Jake Holland, John Kaippallimalil and Vincent Roca for their careful reviews.

Authors' Addresses

Xinpeng Wei
Beiqing Rd. Z-park No.156, Haidian District,
Beijing, 100095, P. R. China
EMail: weixinpeng@huawei.com

Yizhou Li
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China

Phone: +86-25-56624584
EMail: liyizhou@huawei.com

Sami Boutros
VMware, Inc.
EMail: boutross@vmware.com

Liang Geng
China Mobile
EMail: gengliang@chinamobile.com

Transport Area Working Group
Internet-Draft
Intended status: Experimental
Expires: October 23, 2017

J. Holland
Akamai Technologies, Inc.
April 21, 2017

Circuit Breaker Assisted Congestion Control (CBACC): Protocol
Specification
draft-jholland-cb-assisted-cc-01

Abstract

This document specifies Circuit Breaker Assisted Congestion Control (CBACC), which provides bandwidth information from senders to intermediate network nodes to enable good decisions for fast-trip Network Transport Circuit Breaker activity ([I-D.ietf-tsvwg-circuit-breaker]) when necessary for network health. CBACC is specifically designed to support protocols using IP multicast, particularly as a supplement to receiver-driven congestion control protocols to help affected networks rapidly detect and mitigate the impact of scenarios in which a network is oversubscribed to flows which are not responsive to congestion.

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 23, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	4
3. Rationale	4
4. Applicability	5
5. Protocol Specification	5
5.1. Overview	5
5.2. Packet Header Fields	6
5.2.1. Bandwidth Advertisement	6
5.2.1.1. As an IP header option	6
5.2.1.2. Field definitions	7
5.3. States	8
5.3.1. Interface State	8
5.3.2. Flow State	9
5.4. Functionality	10
6. Requirements from other building blocks	12
7. IANA Considerations	12
8. Security Considerations	13
8.1. Forged Packets	13
8.2. Overloading of Slow Paths	14
8.3. Overloading of State	14
9. Acknowledgements	15
10. References	15
10.1. Normative References	15
10.2. Informative References	16
Appendix A. Overjoining	18
Author's Address	19

1. Introduction

This document specifies Circuit Breaker Assisted Congestion Control (CBACC).

CBACC is a congestion control building block designed for use with IP traffic that has a known maximum bandwidth, which does not reduce its sending rate in response to congestion. CBACC is specifically designed to supplement protocols using receiver-driven multicast congestion control systems that rely on well-behaved receivers to achieve congestion control in a very highly scalable system (up to millions of receivers) without a feedback path that reduces sending

rates by senders. Examples of congestion control systems fitting this description include PLM, RLM, RLC, FLID-DL, SMCC, ESMCC, QIRLM, and WEBRC [RFC3738].

CBACC addresses a vulnerability to "overjoining", a condition in which receivers (particularly malicious receivers) subscribe to traffic which, from the sending side, is non-responsive to congestion. Overjoining attacks and the challenges they present are discussed in more detail in Appendix A.

A careful reading of the congestion control requirements of UDP Best Practices [I-D.ietf-tsvwg-rfc5405bis] suggests that a network that forwards multicast traffic is required to operate a circuit breaker to maintain network health under a persistent overjoining condition, at a cost of cutting off some or all multicast traffic across the network during high congestion.

CBACC provides a mechanism for networks to mitigate the impact of overjoining within a network by introducing a mechanism for communicating the bandwidth of non-responsive flows from the sender of the flow to the transit nodes forwarding the flow. The bandwidth information is sufficient to implement a fast-trip circuit breaker [I-D.ietf-tsvwg-circuit-breaker] within a single network node which can specifically block or police flows when receivers have overjoined the network's capacity.

In conjunction with receiver counts (e.g. via [RFC6807]) such nodes can also provide much improved network fairness for circuit breaking decisions during an overjoining condition.

In addition to streams using multicast receiver-driven congestion control, CBACC may also be suitable for use with other traffic, both unicast and multicast, that does not respond to congestion by reducing sending rates, including certain profiles of RTP [RFC3550] over either unicast or multicast, as well as several tunneling protocols (e.g. AMT [RFC7450] and GRE [RFC2784]) when they are known to carry traffic that would be suitable for CBACC. A complete specification for use of CBACC with unicast protocols and with tunneling protocols is out of scope for this document, though the security issues section does mention a few special considerations for potential unicast usage.

CBACC-compliant senders transmit Bandwidth Advertisements through the same transport path as the data traffic, so that circuit breakers can make informed decisions about how flows should be prioritized for circuit breaking. Additionally, CBACC-compliant circuit breakers transmit information to receivers about flows which have been or might soon be circuit-broken, to encourage CBACC-aware applications

to use alternate methods to retrieve equivalent (though probably lower-quality and possibly less efficient) data when possible.

This document describes a building block as defined in [RFC3048]. This document describes a congestion control building block that conforms to [RFC2357]. This document follows the general guidelines provided in [RFC3269], in addition to the requirements on RFCs from [RFC5226] and [RFC3552].

2. Terminology

Term	Definition
circuit breaker	See [I-D.ietf-tsvwg-circuit-breaker]
controlled environment	See [I-D.ietf-tsvwg-rfc5405bis] Section 3.6
general internet flow	See [I-D.ietf-tsvwg-rfc5405bis] Section 3.6
upstream	traffic for a single (source,destination) IP pair, including destinations that are group addresses along a network topology path in the direction of a flow's sender
downstream	along a network topology path in the direction of a flow's receiver
ingress interface	the (single) upstream interface for a flow in a circuit breaker
egress interface	a downstream interface for a flow in a circuit breaker

Table 1

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. Rationale

CBACC is defined as an independent congestion control building block because it would be a useful supplement a wide variety of receiver-driven multicast congestion control schemes, such as [PLM] or other methods based on receiver-driven conformance to a measurement of available network bandwidth or congestion.

CBACC is also potentially valuable, even without other congestion control systems, in controlled environments where congestion control

may not be required (e.g. for certain profiles of RTP [RFC3550]), since CBACC can provide protection for such a network against congestion due to sender or network mis-configuration.

CBACC provides a new form of communication between senders and network transit nodes to facilitate fast-trip circuit breakers as described in section 5.1 of [I-D.ietf-tsvwg-circuit-breaker] which are not available via previously existing methods. When used in conjunction with compatible circuit breakers, CBACC can greatly improve the safety of a network that accepts and delivers interdomain massively scalable multicast traffic to potentially untrusted receivers.

4. Applicability

CBACC relies on the presence of CBACC-aware circuit breakers on a flow's transit path in order to provide congestion control in a network. In the absence of any CBACC-aware circuit breakers on a network path, CBACC constitutes a small extra overhead to a flow without providing any additional value.

CBACC provides a form of congestion control for massively scalable protocols using the IP multicast service. CBACC is best used in conjunction with another receiver-driven multicast congestion control, but it is also suitable for use even without another congestion control mechanism, or when presence of another congestion control mechanism is unproven, such as when accepting multicast joins from untrusted receivers.

5. Protocol Specification

5.1. Overview

CBACC senders send Bandwidth Advertisement packets to advertise the maximum sending bandwidth along the data path for a flow through a network.

CBACC bandwidth information is monitored by CBACC circuit breakers along the network path, which may block the forwarding of traffic for some flows in order to maintain network health. When a flow is blocked, a CBACC circuit breaker sets a bit in Bandwidth Advertisement packets before they're forwarded downstream that indicates to subscribed receivers of that flow that traffic has been blocked.

The protocol also defines a way to notify downstream receivers when a flow is in danger of being circuit broken in the near future. A CBACC-capable transport node SHOULD send this information when it is

known, as described in section [TBD]. This gives applications an opportunity to gracefully shift to a lower-bandwidth version of the same content, when possible, providing an early warning system for avoiding congestion more smoothly.

A Bandwidth Advertisement packet constitutes an "ingress meter" as described in section 3.1 of [I-D.ietf-tsvwg-circuit-breaker]. The configured bandwidth caps of egress interfaces likewise constitute "egress meters". However, the diagram in the referenced document is simplified by running the ingress and egress on the same network node. At the CBACC-aware circuit breaker, the CBACC node has both pieces of information as soon as a Bandwidth Advertisement is received, and can trip the circuit breaker if the aggregate advertised CBACC bandwidth exceeds the actual bandwidth available on any egress interfaces.

5.2. Packet Header Fields

5.2.1. Bandwidth Advertisement

5.2.1.1. As an IP header option

Bandwidth advertisements can appear as either an IPv4 header option (as in Section 3.1 of [RFC0791]) or as an IPv6 extension header option (as in section 4.2 of [RFC2460]). They have the same layout:

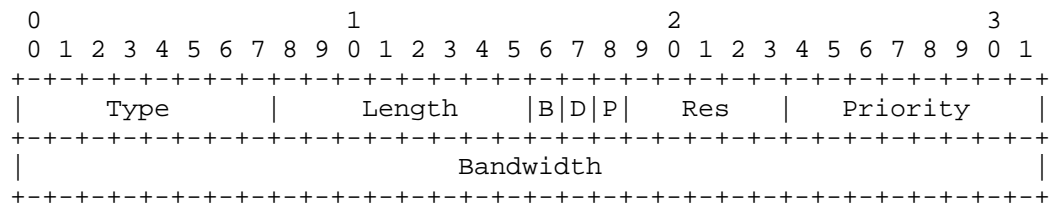


Figure 1

Bandwidth advertisements sent as IPv4 header options use option value [TBD], with the "copied" bit set and the option class "control", as specified in [RFC0791] section 3.1. Until and unless IANA assigns a value, this will be option number 158 as described in section 8 of [RFC4727] for experiments using IPv4 Option types. The length field is 8.

Bandwidth advertisements sent as IPv6 header options use option value [TBD], with the "action" bits set to "skip" and the "change" bit set to 1, as specified in [RFC2460] section 4.2. Until and unless IANA assigns a value, this will be option number 0x3e as described in

section 8 of [RFC4727] for experiments using IPv6 Option Types. The length field is 6.

Using an IP header option has the benefit of exposing the bandwidth to all CBACC-compatible routers, in much the same way the IP Router Alert option would, but without being processed or causing undue load in non-CBACC routers.

The IP Header encapsulations DO work with IPSEC. As described in Appendix A of [RFC4302], the IP header fields are properly treated as mutable and zeroed for the IPSEC ICV calculations. CBACC circuit breakers MAY change bits in transit. The Bandwidth Advertisement header itself IS NOT protected by IPSEC security services, but protection of other parts of the packet remain unchanged.

5.2.1.2. Field definitions

5.2.1.2.1. Bandwidth

As in several other protocols sending bandwidth values such as OSPF-TE [RFC3630], the bandwidth is expressed in bytes per second (not bits), in IEEE floating point format. For quick reference, this format is as follows:

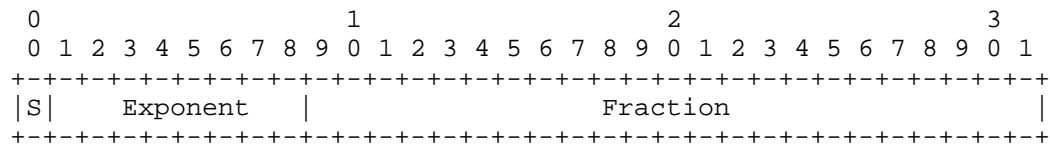


Figure 2

S is the sign, Exponent is the exponent base 2 in "excess 127" notation, and Fraction is the mantissa - 1, with an implied binary point in front of it. Thus, the above represents the value:

$$(-1)^{(S)} * 2^{(Exponent-127)} * (1 + Fraction)$$

For more details, refer to [IEEE.754.1985].

Figure 3

5.2.1.2.2. B (Blocked) bit

Indicates that the flow has been circuit-broken.

5.2.1.2.3. D (Danger) bit

Indicates that the flow is in danger of being circuit-broken.

5.2.1.2.4. P (Police) bit

Indicates that the flow should be policed instead of blocked. Flows marked for policing by the sender should have traffic proportionally dropped when bandwidth is needed, according to their priority. [TBD] Flesh this concept out, and decide whether it's actually viable. This was my attempt at addressing a suggestion from Bob Briscoe at IETF 97 in ICCRG at the mic, IIRC. It probably requires more state, such as total desired policable bandwidth, total current policed bandwidth, and current policing bandwidth per-flow, plus some definition of how to decide between cutting off some flows and policing others. This may not be worth the hassle, but there are some use cases such as FEC repair traffic which might actually be nicer this way. However, it might also be possible to get the same effect by assigning priority to those repair flows. Things like video enhancement layers of course are probably better done as a complete cutoff.

5.2.1.2.5. Res (Reserved bits)

The sender MUST set all reserved bits to 0 when sending a CBACC control packet. Receivers and CBACC-capable transit nodes MUST accept any value in the reserved bits.

5.2.1.2.6. Priority

The sender MAY indicate relative priorities of different streams from the same sender with this field. This is an 8-bit unsigned integer, and higher values are kept preferentially over other traffic from the same sender with lower priority values, so all flows with a lower priority value are circuit-broken before any flows with a higher priority value. Among multiple flows from the same sender with the same priority, the highest bandwidth flows are circuit- broken first.

5.3. States

5.3.1. Interface State

A CBACC circuit breaker holds the following state for each interface, for both the inbound and outbound directions on that interface:

- o aggregate bandwidth: The sum of the bandwidths of all non-circuit-broken CBACC flows which transit this interface in this direction.

- o bandwidth limit: The maximum aggregate CBACC advertised bandwidth allowed, not including circuit-broken flows. This may depend on administrative configuration and congestion measurements for the network, whether from this node or other nodes. It's out of scope for this document to define such congestion measurements. Network operators should carefully consider that this bandwidth limit applies to flows that are unresponsive to congestion.

When reducing the bandwidth limit due to congestion, the circuit breaker **MUST NOT** reduce the limit by more than half its value in 10 seconds, and **SHOULD** use a smoothing function to reduce the limit gradually over time.

It is **RECOMMENDED** that no more than half the capacity for a link be allocated to CBACC flows if the link might be shared with TCP or other traffic that is responsive to congestion.

Depending on administrative configuration and the physical characteristics of the interface, the bandwidth limit may be either shared between upstream and downstream traffic, or it may be separate. Either a single shared value should be used, or two separate independent values should be used for the inbound and outbound directions for an interface.

- o CBACC bandwidth warning threshold: A soft bandwidth threshold. When the aggregate CBACC advertised bandwidth exceeds this threshold, flows that would have been circuit-broken with a bandwidth limit at this threshold **MUST** have the Danger bit set in the Bandwidth Advertisement packets that are forwarded by this circuit breaker. This threshold **SHOULD** be configurable as a proportion of the bandwidth limit, and **MUST** remain at or below the bandwidth limit when the bandwidth limit changes. The recommended proportion value is .75, but specific networks may use a different value if deemed useful by the network operators.

5.3.2. Flow State

The following state is kept for flows that are joined from at least one downstream interface and for which at least one CBACC Bandwidth Advertisement packet has been received:

- o bandwidth: The bandwidth from the most recently received Bandwidth Advertisement.
- o ingress status: One of the following values:
 - * 'subscribed'

Indicates that the circuit breaker is subscribed upstream to the flow and forwarding data and control packets through zero or more egress interfaces.

* 'pruned'

Indicates that the flow has been circuit-broken. A request to unsubscribe from the flow has been sent upstream, e.g. a PIM prune (section 3.5 of [RFC7761]) or a "leave" operation via IGMP, MLD, or another appropriate group membership protocol.

* 'probing'

Indicates that the flow was circuit-broken previously, and is currently joined upstream to refresh the most recent Bandwidth Advertisement in order to evaluate reinstating the flow.

- o probe timer: Used to periodically probe a flow in the 'pruned' state, to evaluate returning to 'forwarding'.

Flows additionally have a per-interface state for egress interfaces:

- o egress status: One of the following values:

* 'forwarding'

Indicates that the flow is a non-circuit-broken flow in steady state, forwarding data and control packets downstream.

* 'blocked'

Indicates that data packets for this flow are NOT forwarded downstream via this interface. Bandwidth Advertisements are still forwarded, each with the 'Blocked' bit set to 1. All other flow traffic MUST be dropped.

5.4. Functionality

The CBACC building block on a sender MUST have access to the maximum bandwidth that may be sent at any time in the following 3 seconds. A CBACC sender MUST send this value in a Bandwidth Advertisement packet once per second. The end result of the traffic sent on the wire for a particular flow MUST honor this maximum bandwidth commitment, such that bandwidth measurements taken over any sliding window one-second period MUST NOT exceed any of prior 3 maximum Bandwidth Advertisements (or any of them, if fewer than 3 have been sent).

A CBACC circuit breaker MUST order its monitored flows based on per-flow estimates of network fairness and preferentially circuit break less fair flows when bandwidth limits are exceeded. A normative method to determine network fairness for a flow is out of scope for this document, but CBACC circuit breaker implementations SHOULD

provide a capability for network operators to configure administrative biases for specific sets of flows, and network operators SHOULD consider fairness concerns as expressed in [RFC2914] section 3.2 and other relevant documents describing best practices.

In particular, fairness metrics SHOULD favor multicast flows with many receivers over multicast flows with few receivers and flows with low bandwidth over flows with high bandwidth. When receiver counts are known (for example via the experimental PIM extension specified in [RFC6807]) a RECOMMENDED metric is (bandwidth/receiver count), though other metrics MAY be used where deemed appropriate by network operators following internet best practices, or when receiver counts can't be determined.

A CBACC sender MUST send Bandwidth Advertisements once per second. (Implementation-specific jitter in timer implementations not exceeding .1s is acceptable.)

If a circuit breaker receives more than 5 Bandwidth Advertisement packets for a flow in two seconds, the circuit breaker SHOULD set the flow to "pruned" and leave the upstream channel, and MUST drop Bandwidth Advertisement packets in excess of one per second.

Flows which are currently circuit-broken on an egress interface are set to "blocked". When a flow on an egress interface is in blocked state, Bandwidth Advertisement packets MUST be forwarded except as described in the preceding paragraph, the "Blocked" bit MUST be set to 1 before forwarding, and other traffic for that flow MUST NOT be forwarded along that interface.

When a flow is blocked or pruned, the circuit breaker MAY truncate the Bandwidth Advertisement packet, keeping only the headers of the packet containing the Bandwidth Advertisement before forwarding.

When a flow is pruned, the circuit-breaker MUST generate and forward a Bandwidth Advertisement packet once per second with the "Blocked" bit set when there are still downstream receivers connected.

In flows which are not circuit-broken but which would be circuit-broken if the bandwidth warning threshold were the bandwidth limit, the Danger bit MUST be set to 1 before forwarding. Both data and control packets are forwarded for flows in this situation. The "Danger" bit MAY be used by receivers to take early action to avoid getting circuit-broken by shifting to a lower-bandwidth representation, if available.

When a flow is in the "blocked" state on every egress interface, the circuit breaker MAY set the flow to "pruned" on the ingress interface and leave the channel upstream.

In addition to monitoring the advertised bandwidth, a CBACC circuit breaker or other assisting nodes in the network SHOULD monitor the observed bandwidth per flow, and SHOULD circuit break "overactive" flows, defined as those which exceed their CBACC maximum bandwidth commitment. A circuit breaker MAY perform constant monitoring on all flows, or MAY use load sharing techniques such as random selection or round robin to monitor only a certain subset of flows at a time.

When detecting overactive flows, circuit breakers MUST use techniques to avoid false positives due to transient upstream network conditions such as packet compression or occasional packet duplication. For example, using an average of bandwidth measurements over the prior 3 seconds would qualify, where a half-second window would not. (A full listing of reasonable false-positive avoidance techniques is out of scope for this document.)

[TBD: examples with network diagrams and bandwidths?] [TBD: some internal structure on this section. "wall of text" was some feedback]

6. Requirements from other building blocks

The sender needs to know the bandwidth, including any upcoming changes, at least 3 seconds in advance. There is no requirement on how building blocks define this functionality except on the packets on the wire--the advance knowledge might, for example, be implemented by buffering and pacing on the sending machine. Specifics of the sending bandwidth implementations are out of scope for this document, as it's intended to provide requirements that will be applicable to a broad range of possible implementations, including RTP and WEBRC.

7. IANA Considerations

This draft requests IANA to allocate an IPv6 packet header option number with the "action" bits set to "skip" and the "change" bit set to 1, as specified in [RFC2460] section 4.2. [TO BE REMOVED: This registration should take place at the following location: <http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#extension-header>.]

This draft also requests IANA to allocate an IPv4 packet header option number with the "copied" bit set and the option class "control", as specified in [RFC0791] section 3.1. [TO BE REMOVED: This registration should take place at the following location:

<http://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml#ip-parameters-1.>]

If those are deemed unacceptable, as an alternative with some compromises described in Section 5.2.1, this draft instead requests IANA to allocate a UDP destination port number. [TO BE REMOVED: This registration should take place at the following location: <http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>.]

8. Security Considerations

8.1. Forged Packets

Forged Bandwidth Advertisement packets that get accepted by CBACC circuit breakers which dramatically over-report or under-report the correct bandwidth would present a potential DoS against a CBACC flow, by making the circuit breaker believe the flow exceeds the node's capacity when over-reporting, or by letting the node notice an apparent violation of the commitment to remain under the advertised bandwidth when under-reporting.

Similarly, it is possible to forge a CBACC Bandwidth Advertisement for a non-CBACC flow, which likewise may constitute a DoS against that flow.

For multicast, attacker would have to be on-path in order to deliver a forged packet to a CBACC circuit breaker, because the join's reverse path propagation will only reach the sender on a legitimate network path to its source address.

For unicast, it's a bigger problem, because ANY sender along path that doesn't have RPF check BCP 38 [RFC2827] permits attack on the flow via forged packet that substantially under-reports or over-reports bandwidth.

For AMT tunnels, when RPF checks along a path to the gateway are not present, nothing stops forged packets from being forwarded by the gateway. If these packets contain CBACC control packets, it's possible to inject a forged packet into the network downstream from the gateway, combining the unicast hole with the multicast hole. This is a vulnerability that should probably be addressed by a new AMT version with some defense against forgery of data.

For IPSEC, since the Bandwidth Advertisement IP header option is mutable, it's not protected by the IPSEC security services, so the Bandwidth Advertisement can be forged for consumption by the circuit breakers, even though the packet will be rejected by the end host

with the security association. This could mount a DoS via the intermediate circuit-breakers by over-reporting or under-reporting flow bandwidth, when processing CBACC traffic through untrusted network paths.

The unicast vulnerabilities would be much mitigated by RPF checks as recommended by BCP 38 [RFC2827] at every hop, or otherwise maintained by the network. Absent such checks, cheap DoS vulnerabilities may be present from any permissive network locations.

8.2. Overloading of Slow Paths

CBACC control packets are sent as part of the data stream so that they traverse the same intermediate network nodes as the rest of the data, but they also carry control information that must be processed by certain nodes along that path.

This creates potential problems very similar to the problems with the Router Alert IP option discussed in Section 3 of [RFC6398], where a circuit-breaker might have a "fast path" for forwarding that can handle a much higher traffic volume than the "slow path" necessary to process CBACC control packets, which is potentially vulnerable to overloading.

If a CBACC-compatible circuit breaker receives a high rate of CBACC control packets, the circuit breaker **MUST** maintain network health for other flows. A circuit-breaker **MAY** drop all packets, including all CBACC control packets, for a flow in which more than 5 CBACC control packets were received in less than a second. (This number is intended to allow for moderate IP packet duplication and packet compression by upstream routers, while still being slow enough for handling of packets on the slow path.)

8.3. Overloading of State

Since CBACC flows require state, it may be possible for a set of receivers and/or senders, possibly acting in concert, to generate many flows in an attempt to overflow the circuit breakers' state tables.

It is permissible for a network node to behave as a CBACC circuit breaker for some CBACC flows while treating other CBACC flows as non-CBACC, as part of a load balancing strategy for the network as a whole, or simply as defense against this concern when the number of monitored flows exceeds some threshold.

The same techniques described in section 3.1 of [RFC4609] can be used to help mitigate this attack, for much the same reasons. It is

RECOMMENDED that network operators implement measures to mitigate such attacks.

9. Acknowledgements

Many thanks to Devin Anderson and Ben Kaduk for detailed reviews and many great suggestions. Thanks also to Cheng Jin, Scott Brown, Miroslav Kaduk, and Bob Briscoe for their thoughtful contributions.

10. References

10.1. Normative References

- [IEEE.754.1985]
Institute of Electrical and Electronics Engineers,
"Standard for Binary Floating-Point Arithmetic",
IEEE Standard 754, August 1985.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791,
DOI 10.17487/RFC0791, September 1981,
<<http://www.rfc-editor.org/info/rfc791>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302,
DOI 10.17487/RFC4302, December 2005,
<<http://www.rfc-editor.org/info/rfc4302>>.

- [RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <<http://www.rfc-editor.org/info/rfc4727>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<http://www.rfc-editor.org/info/rfc7761>>.

10.2. Informative References

- [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-rfc5405bis]
Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", draft-ietf-tsvwg-rfc5405bis-19 (work in progress), October 2016.
- [PLM] A.Legout, E.W.Biersack, Institut EURECOM, "Fast Convergence for Cumulative Layered Multicast Transmission Schemes", 1999.
- [RFC2357] Mankin, A., Romanow, A., Bradner, S., and V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", RFC 2357, DOI 10.17487/RFC2357, June 1998, <<http://www.rfc-editor.org/info/rfc2357>>.
- [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>>.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<http://www.rfc-editor.org/info/rfc2827>>.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<http://www.rfc-editor.org/info/rfc2914>>.

- [RFC3269] Kermode, R. and L. Vicisano, "Author Guidelines for Reliable Multicast Transport (RMT) Building Blocks and Protocol Instantiation documents", RFC 3269, DOI 10.17487/RFC3269, April 2002, <<http://www.rfc-editor.org/info/rfc3269>>.
- [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>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<http://www.rfc-editor.org/info/rfc3552>>.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<http://www.rfc-editor.org/info/rfc3630>>.
- [RFC4609] Savola, P., Lehtonen, R., and D. Meyer, "Protocol Independent Multicast - Sparse Mode (PIM-SM) Multicast Routing Security Issues and Enhancements", RFC 4609, DOI 10.17487/RFC4609, October 2006, <<http://www.rfc-editor.org/info/rfc4609>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC6398] Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, DOI 10.17487/RFC6398, October 2011, <<http://www.rfc-editor.org/info/rfc6398>>.
- [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>>.
- [RFC7450] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, DOI 10.17487/RFC7450, February 2015, <<http://www.rfc-editor.org/info/rfc7450>>.

Appendix A. Overjoining

[I-D.ietf-tsvwg-rfc5405bis] describes several remedies for unicast congestion control under UDP, even though UDP does not itself provide congestion control. In general, any network node under congestion could in theory collect evidence that a unicast flow's sending rate is not responding to congestion, and would then be justified in circuit-breaking it.

With multicast IP, the situation is different, especially in the presence of malicious receivers. A well-behaved sender using a receiver-controlled congestion scheme such as WEBRC does not reduce its send rate in response to congestion, instead relying on receivers to leave the appropriate multicast groups.

This leads to a situation where, when a network accepts inter-domain multicast traffic, as long as there are senders somewhere in the world with aggregate bandwidth that exceeds a network's capacity, receivers in that network can join the flows and overflow the network capacity. A receiver controlled by an attacker could do this at the IGMP/MLD level without running the application layer protocol that participates in the receiver-controlled congestion control.

A network might be able to detect and defend against the most naive version of such an attack by blocking end users that try to join too many flows at once. However, an attacker can achieve the same effect by joining a few high-bandwidth flows, if those exist anywhere, and an attacker that controls a few machines in a network can coordinate the receivers so they join disjoint sets of non-responsive sending flows.

This scenario will produce congestion in a middle node in the network that can't be easily detected at the edge where the IGMP/MLD join is accepted. Thus, an attacker with a small set of machines in a target network can always trip a circuit breaker if present, or can induce excessive congestion among the bandwidth allocated to multicast. This problem gets worse as more multicast flows become available.

This is a significant barrier to multicast adoption because there is no present defense which does not itself constitute a denial of service attack.

Although the same can apply to non-responsive unicast traffic, network operators can assume that non-responsive sending flows are in violation of congestion control best practices, and can therefore cut off such flows. However, non-responsive multicast senders are likely to be well-behaved participants in receiver-controlled congestion control schemes.

However, receiver controlled congestion control schemes also show the most promise for efficient massive scale content distribution via multicast, provided network health can be ensured. Therefore, mechanisms to mitigate overjoining attacks while still permitting receiver-controlled congestion control are necessary. [TBD: this whole section should be expanded and moved to a separate informational draft]

TBD: network diagram

Figure 4

Author's Address

Jacob Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, Massachusetts 02142
USA

Phone: +1 617 444 3000
Email: jholland@akamai.com
URI: <https://www.akamai.com/>

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

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Definitions and Abbreviations	4
3. Architecture Overview	7
4. Procedural Overview	9
4.1. General	9
4.2. Sender Operation with Sliding Window FEC Codes	9
4.3. Receiver Operation with Sliding Window FEC Codes	12
5. Protocol Specification	14
5.1. General	14
5.2. FEC Framework Configuration Information	15
5.3. FEC Scheme Requirements	15
6. Feedback	15
7. Transport Protocols	16
8. Congestion Control	16
9. Implementation Status	16
10. Security Considerations	16
11. Operations and Management Considerations	17
12. IANA Considerations	17
13. Acknowledgments	17
14. References	17
14.1. Normative References	17
14.2. Informative References	17
Appendix A. About Sliding Encoding Window Management (non Normative)	19

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:

- o this extension does not prevent nor compromise 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;
- o 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 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 $0 < \text{code rate} \leq 1$. 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.

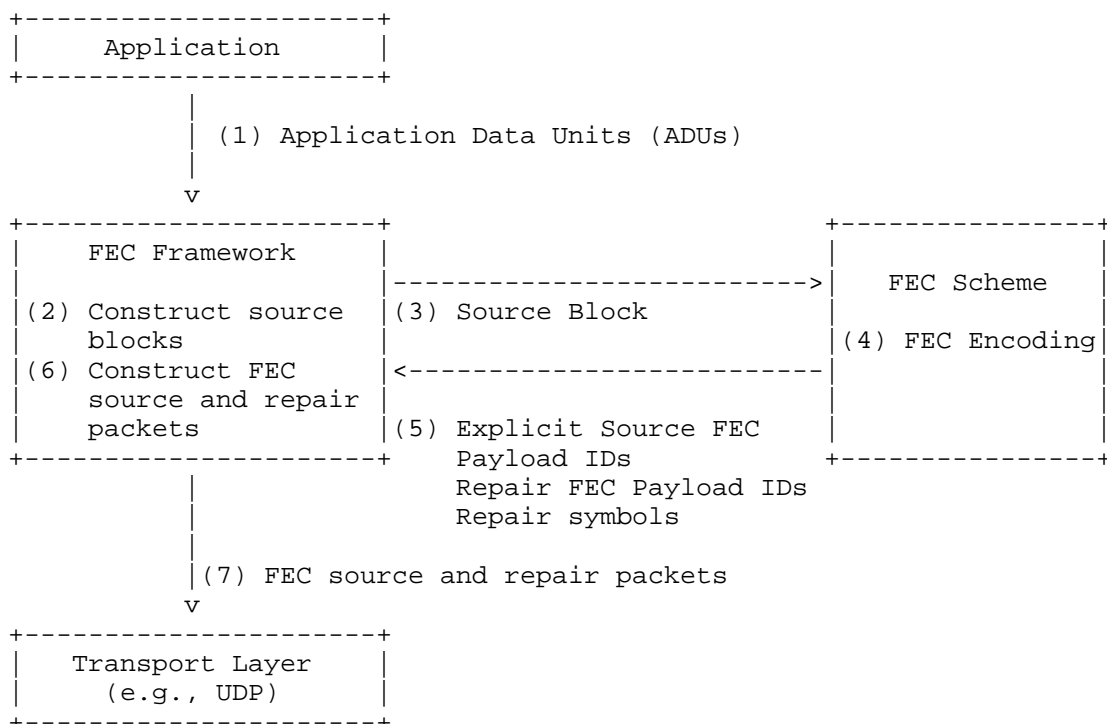


Figure 1: FECFRAME architecture at a sender.

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:

- o 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.
- o 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.

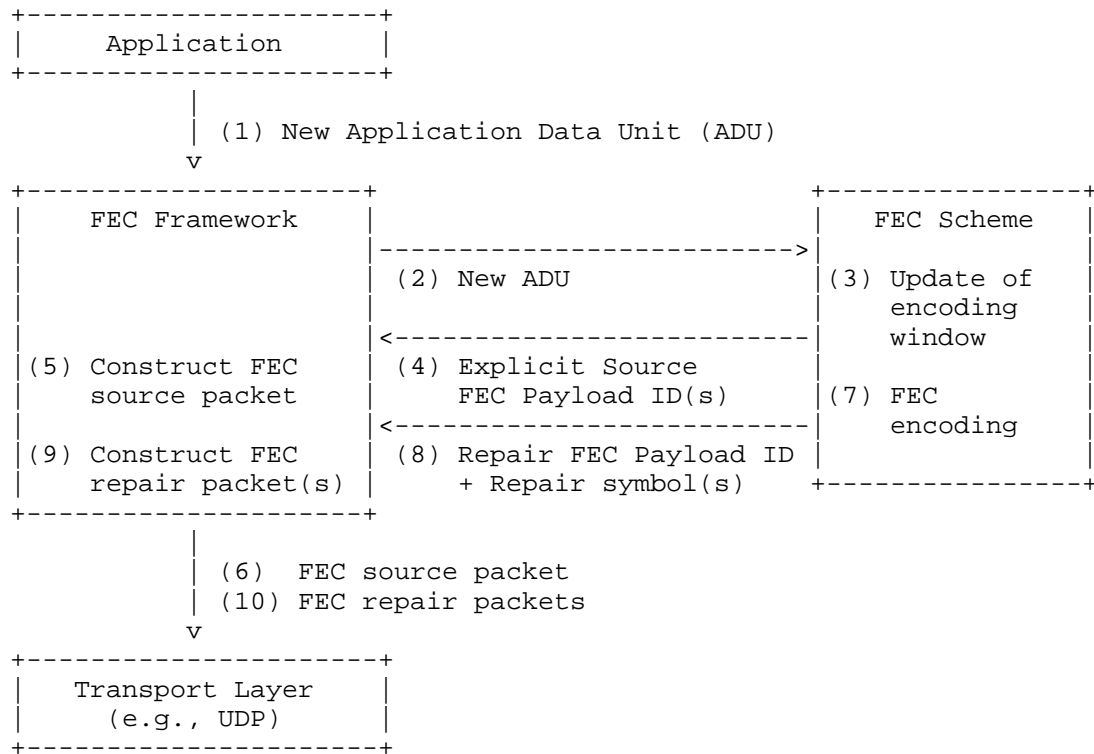


Figure 2: Sender Operation with Convolutional FEC Codes

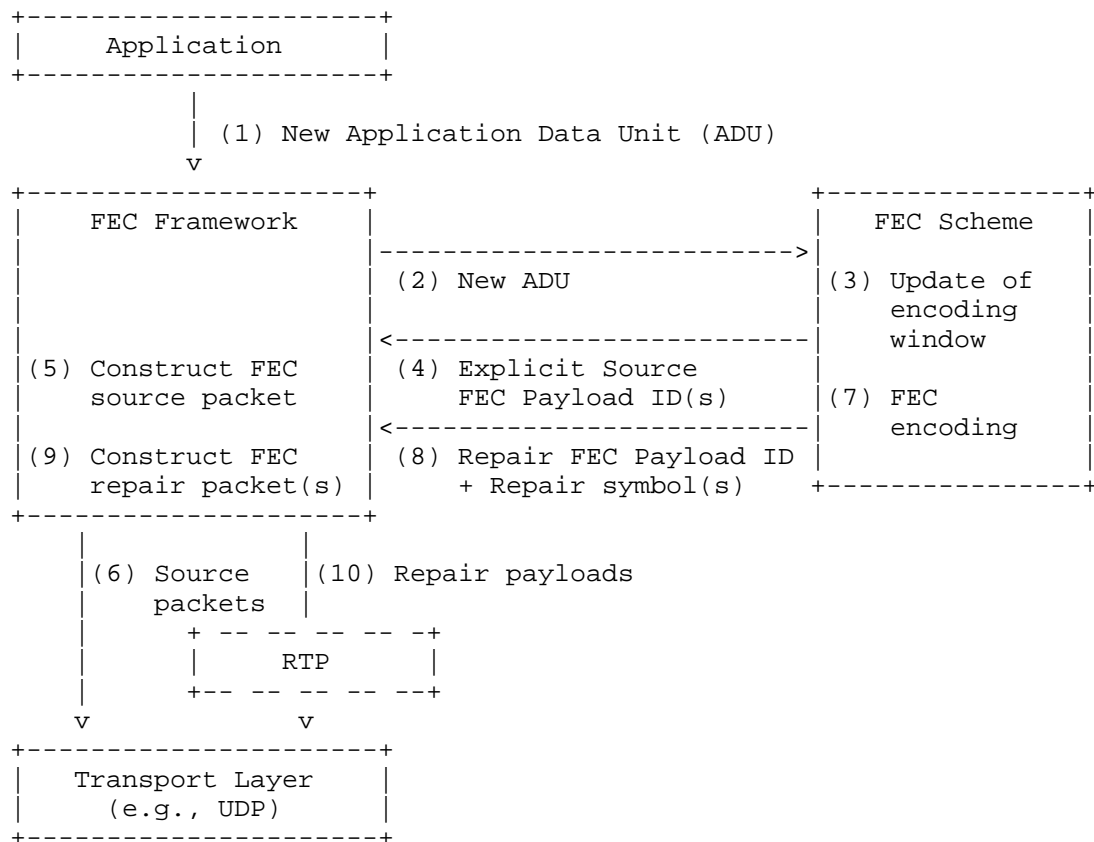


Figure 3: Sender Operation with RTP Repair Flows

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.

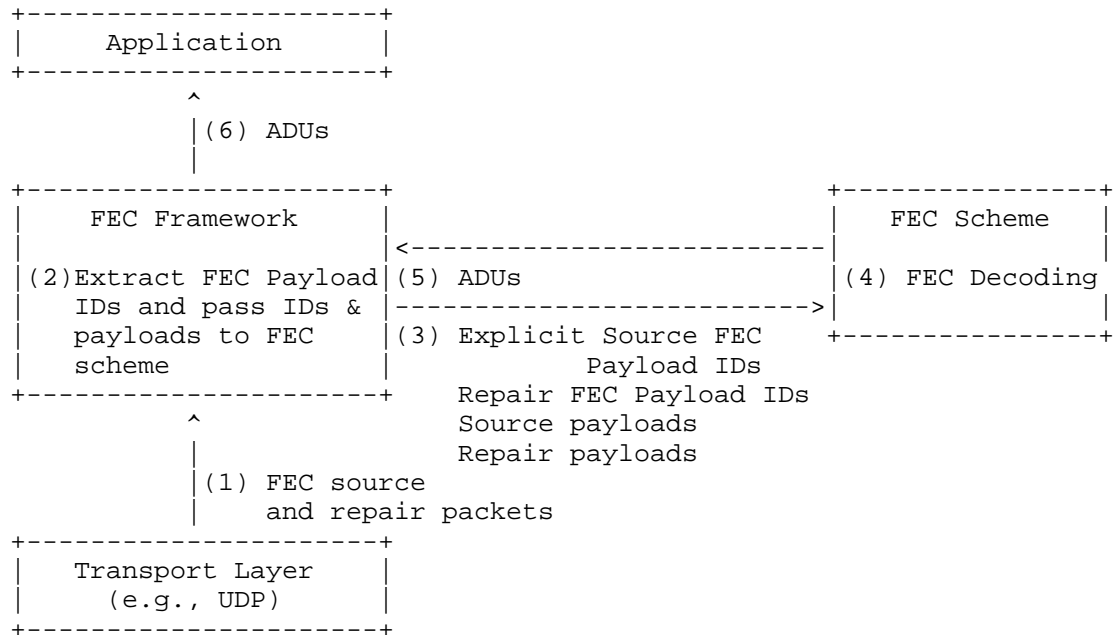


Figure 4: Receiver Operation with Sliding Window FEC Codes

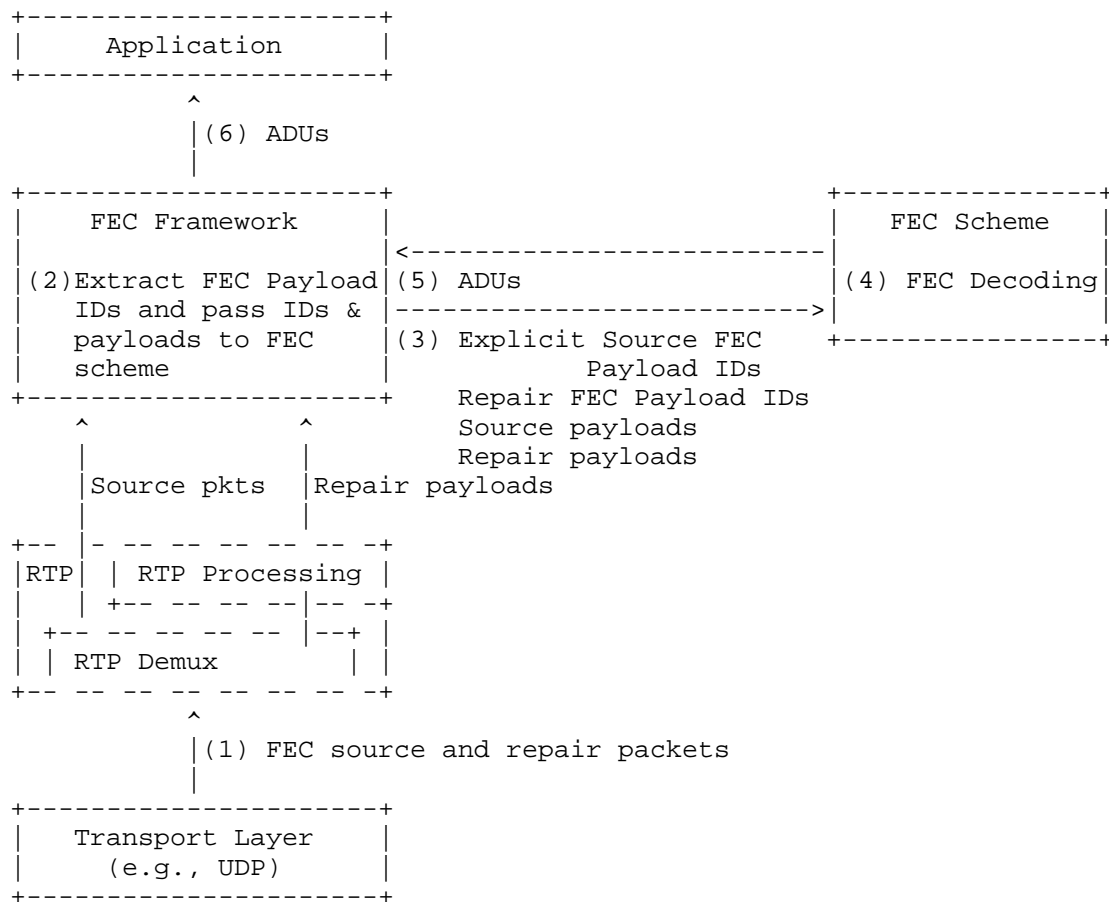


Figure 5: Receiver Operation with RTP Repair Flows

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:

- o MUST define the relationships between ADUs and the associated source symbols (mapping);
- o MUST define the management of the encoding window that slides over the set of ADUs. Appendix A provides a non normative example;
- o 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:

- o Organisation: Inria
- o 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.
- o Maturity: the basic FECFRAME maturity is "production", the FECFRAME extension maturity is "under progress".
- o 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.
- o Lincensing: proprietary.
- o Implementation experience: maximum.
- o Information update date: March 2017.
- o 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:

- o 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).
- o 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:

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

TSGWG
Internet Draft
Intended status: Experimental
Expires: December 2016

J. Touch
USC/ISI
June 28, 2016

Transport Options for UDP
draft-touch-tsvwg-udp-options-03.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on December 28, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

Transport protocols are extended through the use of transport header options. This document experimentally extends UDP to provide a location, syntax, and semantics for transport layer options.

Table of Contents

1. Introduction.....	2
2. Conventions used in this document.....	2
3. Background.....	3
4. The UDP Option Area.....	3
5. Whose options are these?.....	8
6. UDP options vs. UDP-Lite.....	8
7. Interactions with Legacy Devices.....	9
8. Options in a Stateless, Unreliable Transport Protocol.....	10
9. Security Considerations.....	10
10. IANA Considerations.....	11
11. References.....	11
11.1. Normative References.....	11
11.2. Informative References.....	11
12. Acknowledgments.....	12

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 field is typically redundant with the size of the maximum space available as a transport protocol payload (see also discussion in Section 7).

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:

$$\text{UDP_Length} = \text{IPv4_Total_Length} - \text{IPv4_IHL} * 4$$

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:

$$\text{UDP_Length} = \text{IPv6_Payload_Length} - \text{sum}(\text{extension header lengths})$$

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".

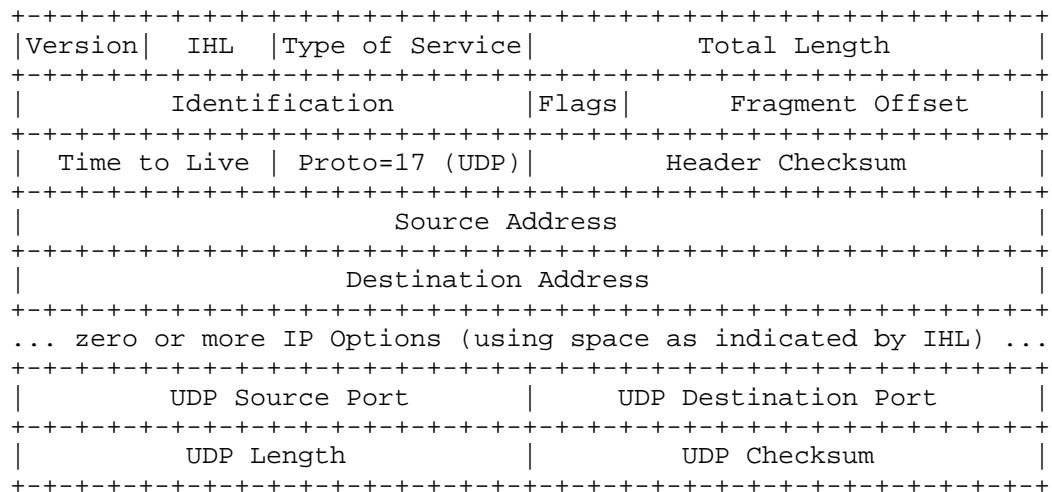


Figure 1 IPv4 datagram with UDP transport payload

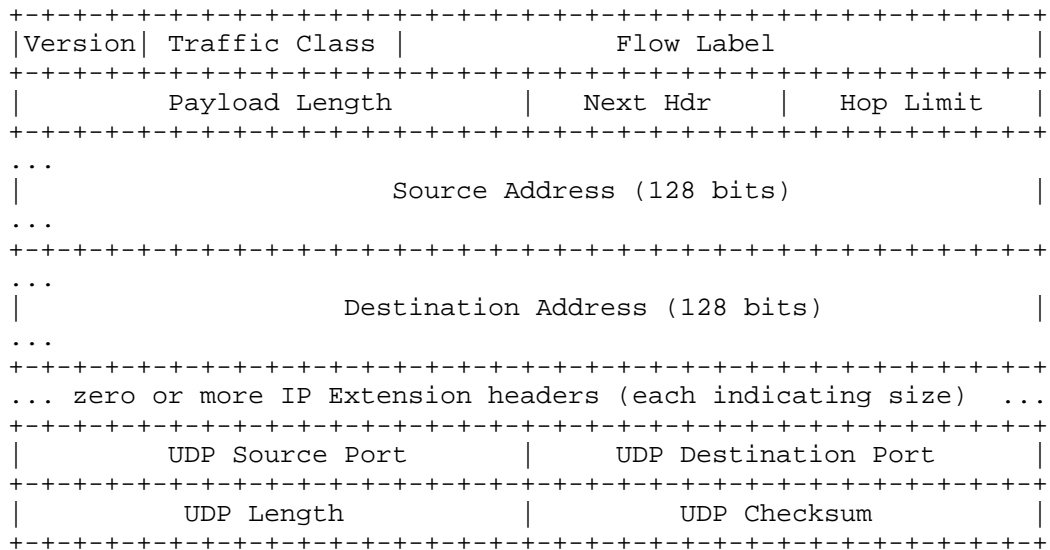


Figure 2 IPv6 datagram with UDP transport payload

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).

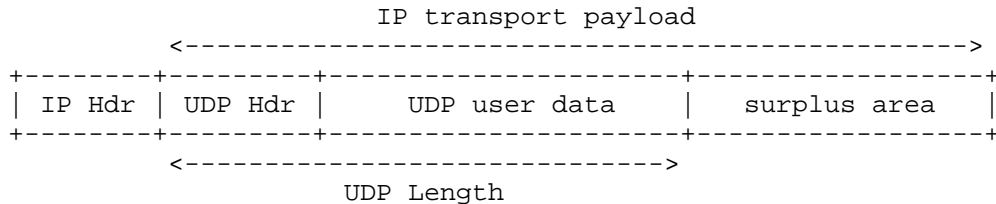


Figure 3 IP transport payload vs. UDP Length

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 7). 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 syntax similar to that of TCP [RFC793]. They are typically a minimum of two bytes in length as shown in Figure 4, excepting only the one byte options "No Operation" (NOP) and "End of Options List" (EOL) described below.

```
+-----+-----+
| Kind  | Length |
+-----+-----+
```

Figure 4 UDP option default format

>> UDP options MAY occur at any UDP length offset.

>> The UDP length MUST be at least as large as the UDP header (8) and no larger than the IP transport payload. Values outside this range MUST be silently discarded as invalid and logged where rate-limiting permits.

Others have considered using values of the UDP Length that is larger than the IP transport payload as an additional type of signal. Using a value smaller than the IP transport payload is expected to be backward compatible with existing UDP implementations, i.e., to deliver the UDP Length of user data to the application and silently ignore the additional surplus area data. Using a value larger than the IP transport payload would either be considered malformed (and be silently dropped) or could cause buffer overruns, and so is not considered silently and safely backward compatible. Its use is thus out of scope for the extension described in this document.

>> UDP options MUST be interpreted in the order in which they occur in the UDP option area.

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)
128-253		RESERVED
254	N(>=4)	RFC 3692-style experiments
255		RESERVED

>> 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.

The option checksum (OCS, Kind = 2) is an 8-bit ones-complement sum that covers only the options, from the first option as indicated by the UDP Length to the last option as indicated by EOL (where present) or the IP Payload Length. 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.

>> When present, the option checksum SHOULD occur as early as possible, preferably preceded by only NOP options for alignment.

>> If the option checksum fails, all options MUST be ignored and any trailing surplus data 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.

>> 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. Those bytes MAY be passed to the application layer if negotiated otherwise in advance; such negotiation can be used as a way to include user data that is not protected by a checksum. If this unprotected data is provided to the user, it MUST be provided distinct from the UDP user data.

Kind=254 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]).

5. 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 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.

6. 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.

7. 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, Mac 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.

There was one embedded device reported that passed the entire IP transport payload to the user UDP socket. This is already inconsistent with UDP and host requirements [RFC768] [RFC1122], as it presents the entire IP transport payload to the user (including the transport header) instead of presenting the transport payload to the corresponding to the transport protocol, where the transport header would have been removed.

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)

8. 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.

It is useful that the above requirements prevent using UDP options to implement transport-layer fragmentation and reassembly 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.

9. 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 a vulnerability and it would rely on inconsistent use of the two fields for memory allocation and copying.

10. 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 indicated herein. 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]. Values in this registry are to be assigned by IANA using first-come, first-served (FCFS) rules [RFC5226].

11. References

11.1. Normative References

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

11.2. Informative References

- [Hi15] Hildebrand, J., B. Trammel, "Substrate Protocol for User Datagrams (SPUD) Prototype," draft-hildebrand-spud-prototype-03, Mar. 2015.
- [RFC768] Postel, J., "User Datagram Protocol", RFC 768, August 1980.
- [RFC791] Postel, J., "Internet Protocol," RFC 791, Sept. 1981.
- [RFC793] Postel, J., "Transmission Control Protocol" RFC 793, September 1981.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -- Communication Layers," RFC 1122, Oct. 1989.
- [RFC2460] Deering, S., R. Hinden, "Internet Protocol Version 6 (IPv6) Specification," RFC 2460, Dec. 1998.
- [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.
- [RFC5925] Touch, J., A. Mankin, R. Bonica, "The TCP Authentication Option," RFC 5925, June 2010.
- [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.
- [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.

12. Acknowledgments

This work benefitted from feedback from Bob Briscoe, Ken Calvert, Ted Faber, Gorrry Fairhurst, C. M. Heard, Tom Herbert, as well as discussions on the IETF SPUD email list.

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

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 is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

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

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

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.....	2
2. Conventions used in this document.....	3
3. Background.....	3
4. The UDP Option Area.....	4
5. UDP Options.....	7
5.1. End of Options List (EOL).....	8
5.2. No Operation (NOP).....	8
5.3. Option Checksum (OCS).....	9
5.4. Alternate Checksum (ACS).....	10
5.5. Lite (LITE).....	10
5.6. Maximum Segment Size (MSS).....	12
5.7. Timestamps (TIME).....	13
5.8. Fragmentation (FRAG).....	13
5.8.1. Coupling FRAG with LITE.....	16
5.9. Authentication and Encryption (AE).....	16
5.10. Experimental (EXP).....	17
6. UDP API Extensions.....	17
7. Whose options are these?.....	18
8. UDP options vs. UDP-Lite.....	18
9. Interactions with Legacy Devices.....	19
10. Options in a Stateless, Unreliable Transport Protocol.....	20
11. UDP Option State Caching.....	20
12. Security Considerations.....	21
13. IANA Considerations.....	22
14. References.....	22
14.1. Normative References.....	22
14.2. Informative References.....	22
15. Acknowledgments.....	24
Appendix A. Implementation Information.....	26

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:

$$\text{UDP_Length} = \text{IPv4_Total_Length} - \text{IPv4_IHL} * 4$$

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:

$$\text{UDP_Length} = \text{IPv6_Payload_Length} - \text{sum}(\text{extension header lengths})$$

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".

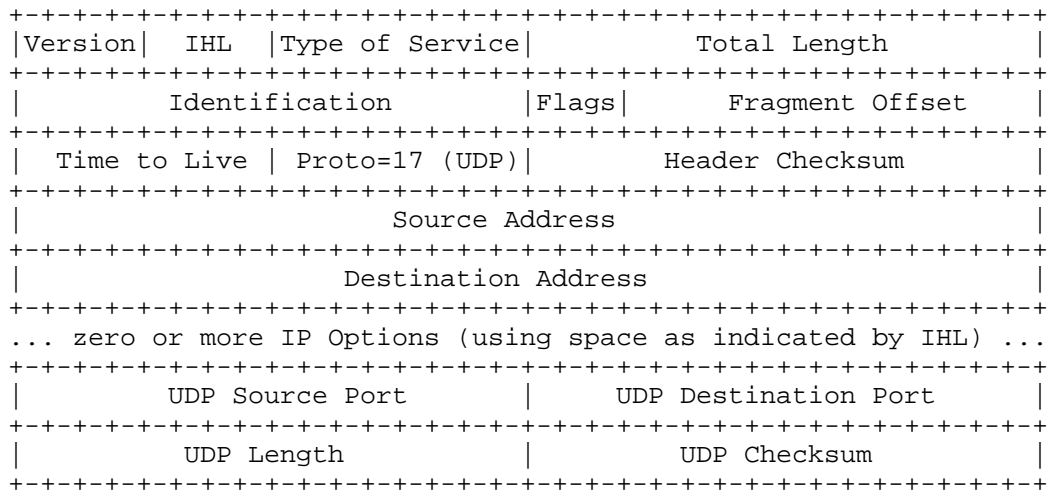


Figure 1 IPv4 datagram with UDP transport payload

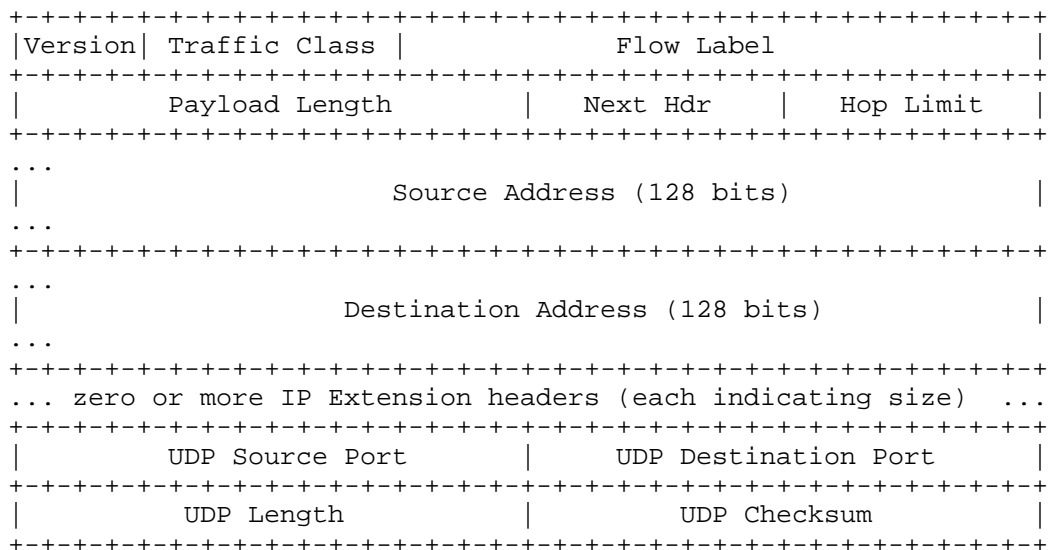


Figure 2 IPv6 datagram with UDP transport payload

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).

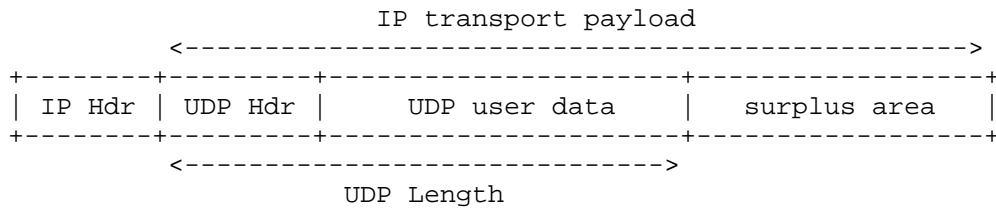


Figure 3 IP transport payload vs. UDP Length

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.

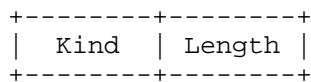


Figure 4 UDP option default format

>> 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.

```
+-----+  
| Kind=0 |  
+-----+
```

Figure 5 UDP EOL option format

>> When the UDP options do not consume the entire option area, the last non-NOP option SHOULD be EOL (vs. filling the entire option area with NOP values).

>> All bytes after EOL MUST be ignored by UDP option processing. As a result, there can only ever be one EOL option (even if other bytes were zero, they are ignored).

5.2. No Operation (NOP)

The No Operation (NOP) option is a one byte placeholder, intended to be used as padding, e.g., to align multi-byte options along 16-bit or 32-bit boundaries.

```

+-----+
| Kind=1 |
+-----+

```

Figure 6 UDP NOP option format

>> If options longer than one byte are used, NOP options SHOULD be used at the beginning of the UDP options area to achieve alignment as would be more efficient for active (i.e., non-NOP) options.

>> Segments SHOULD NOT use more than three consecutive NOPs. NOPs are intended to assign with alignment, not other padding or fill.

[NOTE: Tom Herbert suggested we declare "more than 3 consecutive NOPs" a fatal error to reduce the potential of using NOPs as a DOS attack, but IMO there are other equivalent ways (e.g., using RESERVED or other UNASSIGNED values) and the "no more than 3" creates its own DOS vulnerability)

5.3. Option Checksum (OCS)

The Option Checksum (OCS) is an 8-bit ones-complement sum (Ones8) that covers all of the UDP options. OCS is 8-bits to allow the entire option to occupy a total of 16 bits.

OCS can be calculated by computing the 16-bit ones-complement sum and "folding over" the result (using carry wraparound). Note that OCS is direct, i.e., it is not negated or adjusted if zero (unlike the Internet checksum as used in IPv4, TCP, and UDP headers). OCS protects the option area from errors in a similar way that the UDP checksum protects the UDP user data.

```

+-----+-----+
| Kind=2 | Ones8 |
+-----+-----+

```

Figure 7 UDP OCS option format

>> When present, the option checksum SHOULD occur as early as possible, preferably preceded by only NOP options for alignment and the LITE option if present.

OCS covers the entire UDP option, including the Lite option as formatted before swapping for transmission (or, equivalently, after the swap after reception).

>> If the option checksum fails, all options MUST be ignored and any trailing surplus data (and Lite data, if used) silently discarded.

>> UDP data that is validated by a correct UDP checksum MUST be delivered to the application layer, even if the UDP option checksum fails, unless the endpoints have negotiated otherwise for this segment's socket pair.

5.4. Alternate Checksum (ACS)

The Alternate Checksum (ACS) is a 16-bit CRC of the UDP payload only (excluding the IP pseudoheader, UDP header, and UDP options). It does not include the IP pseudoheader or UDP header, and so need not be updated by NATs when IP addresses or UDP ports are rewritten. Its purpose is to detect errors that the UDP checksum might not detect. CRC-CCITT (polynomial $x^{16} + x^{12} + x^5 + x$ or polynomial 0x1021) has been chosen because of its ubiquity and use in other packet protocols, such as X.25, HDLC, and Bluetooth.

```

+-----+-----+-----+-----+
| Kind=3 | Len=4  |      CRC16sum      |
+-----+-----+-----+-----+

```

Figure 8 UDP ACS option format

5.5. Lite (LITE)

The Lite option (LITE) is intended to provide equivalent capability to the UDP Lite transport protocol [RFC3828]. UDP Lite allows the UDP checksum to cover only a prefix of the UDP data payload, to protect critical information (e.g., application headers) but allow potentially erroneous data to be passed to the user. This feature helps protect application headers but allows for application data errors. Some applications are impacted more by a lack of data than errors in data, e.g., voice and video.

>> When LITE is active, it MUST come first in the UDP options list.

LITE is intended to support the same API as for UDP Lite to allow applications to send and receive data that has a marker indicating the portion protected by the UDP checksum and the portion not protected by the UDP checksum.

LITE includes a 2-byte offset that indicates the length of the portion of the UDP data that is not covered by the UDP checksum.

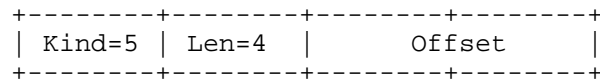


Figure 9 UDP LITE option format

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).

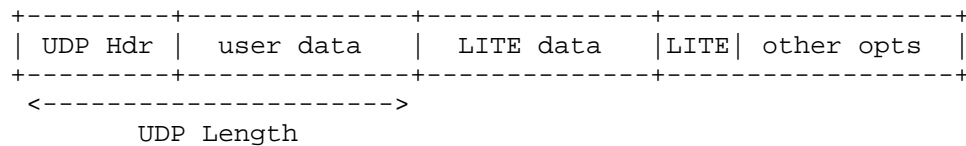


Figure 10 LITE option formation - LITE goes first

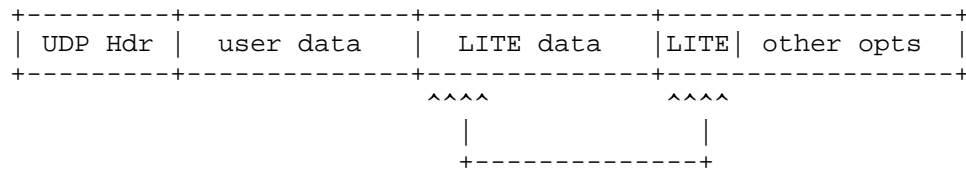


Figure 11 Before sending swap LITE option and front of LITE data

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.

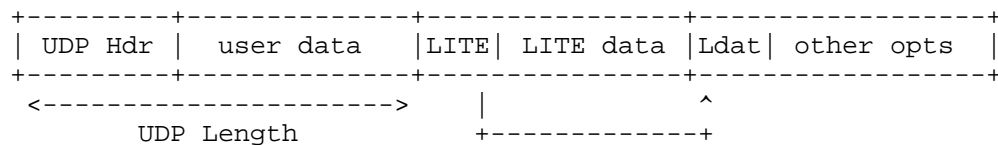


Figure 12 Lite option as sent

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].


```

+-----+-----+-----+-----+
| Kind=5 | Len=4  |     MSS size     |
+-----+-----+-----+-----+

```

Figure 13 UDP MSS option format

The UDP MSS option MAY be used for path MTU discovery [RFC1191][RFC1981], but this may be difficult because of known issues with ICMP blocking [RFC2923] as well as UDP lacking automatic retransmission. It is more likely to be useful when coupled with IP source fragmentation to limit the largest reassembled UDP message, e.g., when EMTU_R is larger than the required minimums (576 for IPv4 [RFC791] and 1500 for IPv6 [RFC2460]).

5.7. Timestamps (TIME)

The UDP Timestamp option (TIME) exchanges two four-byte timestamp fields. It serves a similar purpose to TCP's TS option [RFC7323], enabling UDP to estimate the round trip time (RTT) between hosts. For UDP, this RTT can be useful for establishing UDP fragment reassembly timeouts or transport-layer rate-limiting [RFC8085].

```

+-----+-----+-----+-----+-----+
| Kind=6 | Len=10 |     TS Value     |     TS Echo Reply |
+-----+-----+-----+-----+-----+
      1 byte   1 byte       4 bytes           4 bytes

```

Figure 14 UDP TIME option format

TS Value (TSval) and TS Echo (TSecr) are used in a similar manner to the TCP TS option [RFC7323]. A host using the Timestamp option sets TS Value on all UDP segments issued. Received TSval values are provided to the application, which passes this value as TSecr on UDP messages sent in response to such a message.

>> UDP MAY use an RTT estimate based on nonzero Timestamp values as a hint for fragmentation reassembly, rate limiting, or other mechanisms that benefit from such an estimate.

>> UDP SHOULD make this RTT estimate available to the user application.

5.8. Fragmentation (FRAG)

The Fragmentation option (FRAG) supports UDP fragmentation and reassembly, which can be used to transfer UDP messages larger than limited by the IP receive MTU (EMTU_R [RFC1122]). It is typically

used with the UDP MSS option to enable more efficient use of large messages, both at the UDP and IP layers. FRAG is designed similar to the IPv6 Fragmentation Header [RFC2460], except that the UDP variant uses a 16-bit Offset measured in bytes, rather than IPv6's 13-bit Fragment Offset measured in 8-byte units. This UDP variant avoids creating reserved fields.

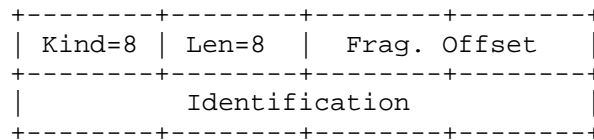


Figure 15 UDP non-terminal FRAG option format

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).

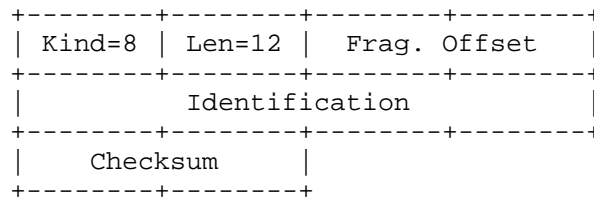


Figure 16 UDP terminal FRAG option format

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.

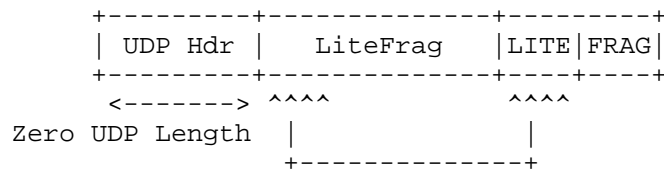


Figure 17 Preparing FRAG as Lite data

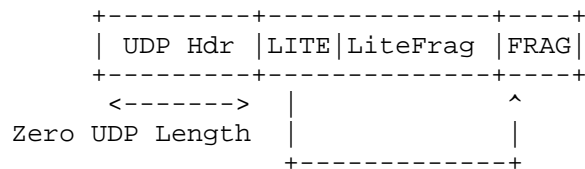


Figure 18 Lite option before transmission

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]:

- o Method to create new receive ports
 - o E.g., `bind(handle, recvaddr(optional), recvport)`
- o Receive, which returns data octets, source port, and source address
 - o E.g., `recvfrom(handle, srcaddr, srcport, data)`
- o Send, which specifies data, source and destination addresses, and source and destination ports
 - o E.g., `sendto(handle, destaddr, destport, data)`

This API is extended to support options as follows:

- o Extend the method to create receive ports to include receive options that are required. Datagrams not containing these required options MUST be silently dropped and MAY be logged.
- o Extend the receive function to indicate the options and their parameters as received with the corresponding received datagram.
- o Extend the send function to indicate the options to be added to the corresponding sent datagram.

Examples of API instances for Linux and FreeBSD are provided in Appendix A, to encourage uniform cross-platform implementations.

7. Whose options are these?

UDP options are indicated in an area of the IP payload that is not used by UDP. That area is really part of the IP payload, not the UDP payload, and as such, it might be tempting to consider whether this is a generally useful approach to extending IP.

Unfortunately, the surplus area exists only for transports that include their own transport layer payload length indicator. TCP and SCTP include header length fields that already provide space for transport options by indicating the total length of the header area, such that the entire remaining area indicated in the network layer (IP) is transport payload. UDP-Lite already uses the UDP Length field to indicate the boundary between data covered by the transport checksum and data not covered, and so there is no remaining area where the length of the UDP-Lite payload as a whole can be indicated [RFC3828].

UDP options are intended for use only by the transport endpoints. They are no more (or less) appropriate to be modified in-transit than any other portion of the transport datagram.

UDP options are transport options. Generally, transport datagrams are not intended to be modified in-transit. However, the UDP option mechanism provides no specific protection against in-transit modification of the UDP header, UDP payload, or UDP option area, except as provided by the options selected (e.g., OCS, ACS, or AE).

8. UDP options vs. UDP-Lite

UDP-Lite provides partial checksum coverage, so that packets with errors in some locations can be delivered to the user [RFC3828]. It uses a different transport protocol number (136) than UDP (17) to

interpret the UDP Length field as the prefix covered by the UDP checksum.

UDP (protocol 17) already defines the UDP Length field as the limit of the UDP checksum, but by default also limits the data provided to the application as that which precedes the UDP Length. A goal of UDP-Lite is to deliver data beyond UDP Length as a default, which is why a separate transport protocol number was required.

UDP options do not need a separate transport protocol number because the data beyond the UDP Length offset (surplus data) is not provided to the application by default. That data is interpreted exclusively within the UDP transport layer.

UDP options support a similar service to UDP-Lite by terminating the UDP options with an EOL option. The additional data not covered by the UDP checksum follows that EOL option, and is passed to the user separately. The difference is that UDP-Lite provides the un-checksummed user data to the application by default, whereas UDP options can provide the same capability only for endpoints that are negotiated in advance (i.e., by default, UDP options would silently discard this non-checksummed data). Additionally, in UDP-Lite the checksummed and non-checksummed payload components are adjacent, whereas in UDP options they are separated by the option area - which, minimally, must consist of at least one EOL option.

UDP-Lite cannot support UDP options, either as proposed here or in any other form, because the entire payload of the UDP packet is already defined as user data and there is no additional field in which to indicate a separate area for options. The UDP Length field in UDP-Lite is already used to indicate the boundary between user data covered by the checksum and user data not covered.

9. Interactions with Legacy Devices

It has always been permissible for the UDP Length to be inconsistent with the IP transport payload length [RFC768]. Such inconsistency has been utilized in UDP-Lite using a different transport number. There are no known systems that use this inconsistency for UDP [RFC3828]. It is possible that such use might interact with UDP options, i.e., where legacy systems might generate UDP datagrams that appear to have UDP options. The UDP OCS provides protection against such events and is stronger than a static "magic number".

UDP options have been tested as interoperable with Linux, Max OS-X, and Windows Cygwin, and worked through NAT devices. These systems

successfully delivered only the user data indicated by the UDP Length field and silently discarded the surplus area.

One reported embedded device passes the entire IP datagram to the UDP application layer. Although this feature could enable application-layer UDP option processing, it would require that conventional UDP user applications examine only the UDP payload. This feature is also inconsistent with the UDP application interface [RFC768] [RFC1122].

It has been reported that Alcatel-Lucent's "Brick" Intrusion Detection System has a default configuration that interprets inconsistencies between UDP Length and IP Length as an attack to be reported. Note that other firewall systems, e.g., CheckPoint, use a default "relaxed UDP length verification" to avoid falsely interpreting this inconsistency as an attack.

(TBD: test with UDP checksum offload and UDP fragmentation offload)

10. Options in a Stateless, Unreliable Transport Protocol

There are two ways to interpret options for a stateless, unreliable protocol -- an option is either local to the message or intended to affect a stream of messages in a soft-state manner. Either interpretation is valid for defined UDP options.

It is impossible to know in advance whether an endpoint supports a UDP option.

>> UDP options MUST allow for silent failure on first receipt.

>> UDP options that rely on soft-state exchange MUST allow for message reordering and loss.

>> A UDP option MUST be silently optional until confirmed by exchange with an endpoint.

The above requirements prevent using any option that cannot be safely ignored unless that capability has been negotiated with an endpoint in advance for a socket pair. Legacy systems would need to be able to interpret the transport payload fragments as individual transport datagrams.

11. UDP Option State Caching

Some TCP connection parameters, stored in the TCP Control Block, can be usefully shared either among concurrent connections or between

connections in sequence, known as TCP Sharing [RFC2140][To17cb]. Although UDP is stateless, some of the options proposed herein may have similar benefit in being shared or cached. We call this UCB Sharing, or UDP Control Block Sharing, by analogy.

[TBD: extend this section to indicate which options MAY vs. MUST NOT be shared and how, e.g., along the lines of To17cb]

Updates to RFC 768

This document updates RFC 768 as follows:

- o This document defines the meaning of the IP payload area beyond the UDP length but within the IP length.
- o This document extends the UDP API to support the use of options.

12. Security Considerations

The use of UDP packets with inconsistent IP and UDP Length fields has the potential to trigger a buffer overflow error if not properly handled, e.g., if space is allocated based on the smaller field and copying is based on the larger. However, there have been no reports of such vulnerability and it would rely on inconsistent use of the two fields for memory allocation and copying.

UDP options are not covered by DTLS (datagram transport-layer security). Despite the name, neither TLS [RFC5246] (transport layer security, for TCP) nor DTLS [RFC6347] (TLS for UDP) protect the transport layer. Both operate as a shim layer solely on the payload of transport packets, protecting only their contents. Just as TLS does not protect the TCP header or its options, DTLS does not protect the UDP header or the new options introduced by this document. Transport security is provided in TCP by the TCP Authentication Option (TCP-AO [RFC5925]) or in UDP by the Authentication Extension option (Section 5.9). Transport headers are also protected as payload when using IP security (IPsec) [RFC4301].

UDP options use the TLV syntax similar to that of TCP. This syntax is known to require serial processing and may pose a DOS risk, e.g., if an attacker adds large numbers of unknown options that must be parsed in their entirety. Implementations concerned with the potential for this vulnerability MAY implement only the required options and MAY also limit NOPs (e.g., no more than three consecutive NOPs or some total number that might occur between the required options, if all are present). Because the required options

come first and at most once each (and all later duplicates silently ignored), this limits the DOS impact.

13. IANA Considerations

Upon publication, IANA is hereby requested to create a new registry for UDP Option Kind numbers, similar to that for TCP Option Kinds. Initial values of this registry are as listed in Section 5. Additional values in this registry are to be assigned by IESG Approval or Standards Action [RFC5226].

Upon publication, IANA is hereby requested to create a new registry for UDP Experimental Option Experiment Identifiers (UDP ExIDs) for use in a similar manner as TCP ExIDs [RFC6994]. This registry is initially empty. Values in this registry are to be assigned by IANA using first-come, first-served (FCFS) rules [RFC5226].

14. References

14.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC768] Postel, J., "User Datagram Protocol", RFC 768, August 1980.
- [RFC791] Postel, J., "Internet Protocol," RFC 791, Sept. 1981.

14.2. Informative References

- [Hil15] 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.
- [Tol7ao] Touch, J., "A TCP Authentication Option Extension for Payload Encryption", draft-touch-tcp-ao-encrypt, Apr. 2017.
- [Tol7cb] 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, Gorrry 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

Internet Engineering Task Force
Internet-Draft
Updates: 3662,4594 (if approved)
Intended status: Standards Track
Expires: April 24, 2017

R. Bless
Karlsruhe Institute of Technology (KIT)
October 21, 2016

A Lower Effort Per-Hop Behavior (LE PHB)
draft-tsvwg-le-phb-00

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 April 24, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Applicability	3
1.2. Deployment Considerations	4
1.3. Requirements Language	4
2. PHB Description	4
3. Traffic Conditioning Actions	5
4. Recommended DS Codepoint	5
5. Remarking to other DSCPs/PHBs	5
6. IANA Considerations	6
7. Security Considerations	6
8. References	6
8.1. Normative References	6
8.2. Informative References	6
Appendix A. History of the LE PHB	7
Appendix B. Acknowledgments	7
Author's Address	7

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:

- o For traffic caused by world-wide web search engines while they gather information from web servers.
- o For software updates or dissemination of new releases of operating systems.

- o For backup traffic or non-time critical sychronization or mirroring traffic.
- o For content distribution transfers between caches.
- o For Netnews and other "bulk mail" of the Internet.
- o 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:

- o 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.
- o No parameters or configuration: the LE PHB requires no parameters and no configuration of traffic profiles and so on.
- o 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 76131
Germany

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

Network Working Group
Internet-Draft
Updates: 6951 (if approved)
Intended status: Standards Track
Expires: May 2, 2017

M. Tuexen
Muenster Univ. of Appl. Sciences
R. Stewart
Netflix, Inc.
October 29, 2016

Additional Considerations for UDP Encapsulation of Stream Control
Transmission Protocol (SCTP) Packets
draft-tuexen-tsvwg-sctp-udp-encaps-cons-01.txt

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 <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 May 2, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions	2
3. Handling of Out of the Blue Packets	2
4. Handling of SCTP Packets Containing an INIT Chunk Matching an Existing Association	3
5. IANA Considerations	5
6. Security Considerations	6
7. Acknowledgments	6
8. Normative References	6
Authors' Addresses	7

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 for the handling of received packets where this verification tag check is not possible. This includes packets for which no association can be found and packets containing an INIT chunk, since the verification tag for these packets must be 0.

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

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

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 bytes the following figure.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Cause Code = 14           |           Cause Length = 8           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Current Encapsulation Port   |   New Encapsulation Port   |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Cause Code: 2 bytes (unsigned integer)

This field MUST hold the IANA defined error cause code for the "Restart of an Association with New Encapsulation Port" error cause. The suggested value of this field for IANA is 14.

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. 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 value for the error cause code is 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]

6. 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.

7. Acknowledgments

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

The authors wish to thank Irene Ruengeler 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).

8. 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>>.
- [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>>.

Authors' Addresses

Michael Tuexen
Muenster 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

Network Working Group	M. Tüxen
Internet-Draft	Münster Univ. of Appl. Sciences
Updates: 6951 (if approved)	R. R. Stewart
Intended status: Standards Track	Netflix, Inc.
Expires: 31 August 2022	27 February 2022

Additional Considerations for UDP Encapsulation of Stream Control
Transmission Protocol (SCTP) Packets
draft-tuexen-tsvwg-sctp-udp-encaps-cons-05

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 of the handling of received packets for which this check is not possible.

This document updates RFC 6951 by specifying the handling of received packets for which 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 31 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Conventions	2
3. Handling of Out of the Blue Packets	3
4. Handling of SCTP Packets Containing an INIT Chunk Matching an Existing Associations	3
5. Middlebox Considerations	5
6. IANA Considerations	5
7. Security Considerations	6
8. Acknowledgments	6
9. Normative References	6
Authors' Addresses	7

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 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 received packet was not encapsulated in UDP, the response packet MUST NOT be encapsulated in UDP.

Please note that in these cases a check 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 tag 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.

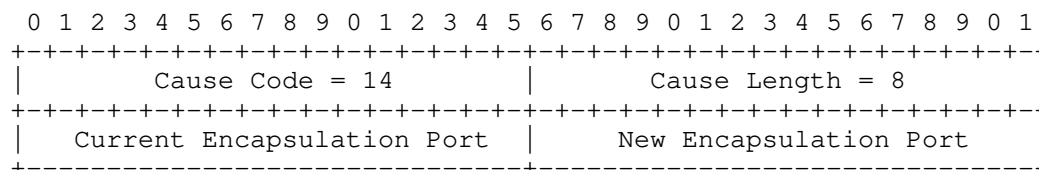


Figure 1: Restart of an Association with New Encapsulation Port error cause

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 cause code 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:

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

Table 1: New entry in Error Cause Codes registry

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.
2455 Heritage Green Ave
Davenport, FL 33837
United States
Email: randall@lakerest.net

INTERNET-DRAFT
Intended Status: Standard Track
Expires: May 4, 2017

Y.Yu
HUAWEI Technologies
October 31, 2016

Layer 3 Quantized Congestion Notification(L3QCN)in the Converged Network
draft-yu-tsvwg-l3qcn-00

Abstract

The more demands for the lossless and low latency network in the modern datacenter appear because the proliferation of demanding applications. Some congestion control schemes such as CN, PFC, ETS which is introduced by IEEE 802.1 focus on the L2 network domain. While current TCP/IP stacks can't meet these requirement on L3 or above networks. This draft introduces the L3QCN(Layer 3 Quantized Congestion Notification), an end to end congestion control scheme which adopt QCN and DCQCN on L2 network. It specifies protocols, procedures, and managed objects to support congestion control on the datacenter network.

Status of this Memo

This Internet-Draft is submitted to IETF 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 May 4, 2017.

Copyright and License Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	3
1.1	Terminology	3
2	Current Congestion Control method	4
2.1	QCN Introduction	4
2.1.1	QCN Technical Solution	4
2.1.2	The limitation of QCN	5
2.2	Introduction of DCQCN	6
2.2.1	DCQCN technical solution	6
2.2.2	The limitation of DCQCN	6
3.	Layer3 QCN	6
3.1	L3QCN Introduction	6
3.2	Use case of L3QCN	6
3.2.1	A hybrid method with QCN	6
3.2.2	L3QCN in CLOS fat-tree	7
4.	Conclusion	11
5	Security Considerations	11
6	IANA Considerations	11
7	References	11
7.1	Normative References	11
7.2	Informative References	11
	Authors' Addresses	12

1 Introduction

Currently, there are 3 classes of streams in the DC network:

- 1)Storage Traffic (Lossless)
- 2)High Compute Traffic(Low latency)
- 3)Ethernet Traffic (Certain packet loss& latency tolerance)

Traditional DC network treat different traffic with different network bearer which exist in the small scale DC. While with the expand of the DC scale, there is an available method which use the Ethernet to bear the streams by applying the congestion control method. IEEE has introduced the following specifications:

1. Enhanced Transmission Selection (ETS) [1] When the offered load in a traffic class doesn't use its allocated bandwidth, enhanced transmission selection will allow other traffic classes to use the available bandwidth. This avoid the burst of one class traffic to influence other classes which provide the minimum guaranteed bandwidth to all traffic classes. This also facilitate the multiple classes exist in one network.

2.Priority-based Flow Control(PFC) [2] Data Center Bridging networks (bridges and end nodes) are characterized by limited bandwidth-delay product and limited hop-count. Traffic class is identified by the VLAN tag priority values. Priority-based flow control is intended to eliminate frame loss due to congestion. This realized the lossless of storage stream and no impact to other 2 traffic classes when all the 3 traffic classes coexist in the Ethernet.

3.Quantized Congestion Notification (QCN) [3] This mechanism enable bridges to signal congestion information to end stations capable of transmission rate limiting to avoid frame loss. Resolve the latency increase caused by flow control or packet retransmission to achieve the higher network throughput.

This draft introduce a L3QCN method to resolve the congestion problem under the converged network in the datacenter. Different classes of traffic will be configured with corresponding priorities. Bridge will apply the policies of congestion control according to the traffic of congested traffic which is defined by the priority.

1.1 Terminology

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

2 Current Congestion Control method

2.1 QCN Introduction

2.1.1 QCN Technical Solution

QCN is defined in IEEE 802.1Qau, there are 2 types of Ethernet frame: One data frame with CN-TAG as Figure 1 shown. The Converged Network Adapters (CNA) which support QCN function will send out the CN-TAG frame when connecting network domain. The difference from the normal frame is the CN-TAG field in the head of Ethernet frame which includes RPID (also known as FLOW-ID). RPID will uniquely identifies every stream sent by the adaptor. When the congestion appeared, bridge will send out CNM frame(introduced in second clause) to notify the source node to stop sending this stream. The FLOW-ID of source frame will be encapsulated in the CNM frame. When the adaptor receives the CNM frame, it will reduce the transmission rate of the identified flow in order to control the specific traffic precisely.

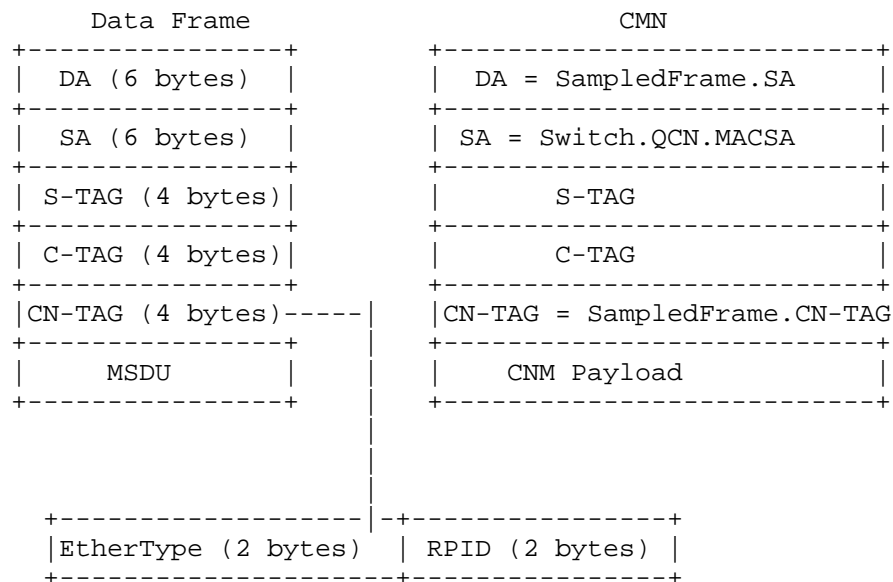


Figure 1. QCN data frame and CMN

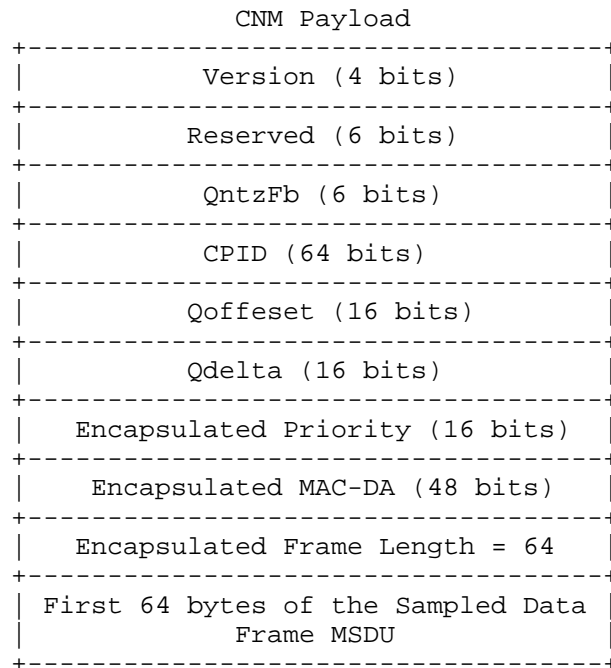


Figure 2. CMN payload

The CNM frame is shown as Figure 2:

- Field 1: Version of CNM message (4 bits)
- Field 2: Reserved (6 bits)
- Field 3: QntzFB, Quantized feedback of CNM message (6 bits)
- Field 4: Congestion Point Identifier (CPID, 8 bytes). In order to assure the uniqueness of the identifier, use the MAC address as the upper 6 bytes. Lower 2 bytes identify the different ports or different priority classes in the same device.
- Field 5: QOffset (2 bytes). Current number of available bytes in the sending queue of the congested point (CP)
- Field 6: QDelta (2 bytes), the difference of available bytes of CP at 2 time point.
- Field 7: Encapsulated priority (2 bytes). Use upper 3 bits of the 1st byte to fill the priority of the CNM frame. Else is 0.
- Field 8: Encapsulated destination MAC address (6 bytes). Fill the destination MAC address which trigger the CNM frame.
- Field 9: Encapsulated MSDU length (2 bytes). The length of the Encapsulated MSDU.
- Field 10: Encapsulated MSDU (64 bytes). Fill in the payload of the CNM.

2.1.2 The limitation of QCN

During the congestion, bridge need to encapsulate the FLOW-ID(in the head of Ethernet frame) in the CNM. Then replace the destination MAC address of CNM with the source MAC address of the congested frame in order to ensure CNM could be send back the sending server. Sending server reduce the flow according to the FLOW-ID carried in the CNM. This characteristic limit QCN only in Ethernet(Level 2 in ISO). Since the head of Ethernet frame will be changed during every packet routing in the IP network, the FLOW-ID and MAC address of sending server will be lost. So the downstream bridge could not create the CNM and send back to the sending server. QCN couldn't support the Layer 3 networking.

2.2 Introduction of DCQCN

2.2.1 DCQCN technical solution

DCQCN[4] is a kind of congestion control solution proposed by Microsoft for the DC network domain. DCQCN is mainly deployed in the RoCEv2 scene. CP (Congestion Point, bridge) set the CN(congestion notification) for the datagram with probability according to the degree of the congestion. After the datagram sent to NP(Notification Point, receiving server), NP construct CNP (Congestion Notification Packet) to RP(Reaction Point, sending server). RP reduce or increase the transmission rate according to the dedicated algorithm which is similar to QCN.

2.2.2 The limitation of DCQCN

DCN construct ECN(explicit congestion notification)[5] tag during the congestion and forward to NP. NP construct CNP to notify RP. The reaction is not quite timely(Control Loop Delay is big). If the congestion appeared on the upper jump, for example on the TOR, there is more delay of 9 jumps than the direct response.

3. Layer3 QCN

3.1 L3QCN Introduction

L3QCN is a technical solution to resolve the congestion problem under the converged network in the datacenter. Different class of traffic will be configured with corresponding priorities. Bridge will deploy the policies of congestion control according to the class of congested traffic which is defined by the priority.

3.2 Use case of L3QCN

3.2.1 A hybrid method with QCN

Deploy priority 5 to the traffic sent out by QCN server. When the queue buffer for the priority 5 exceed the defined threshold, the bridge will back-haul the congestion information to the accessing TOR. TOR and HOST can reach to each other on the Ethernet which is similar to a L2 domain. In this situation, standard QCN is performed. Accessing TOR transform the congestion information to standard CNM frame and send to QCN server

which realize the congestion control.

Deploy priority 7 to the traffic sent out by RoCEv2 server. When the queue buffer for the priority 7 exceed the defined threshold, CP judges the key flow causing the congestion. Then CP construct the standard CNP. The RoCEv2 server reduce the transform rate according to the probability of CNP reception.

3.2.2 L3QCN in CLOS fat-tree

L3QCN control steps are as follows:

1)Datagram sent out from QCN server enters the accessing TOR. Firstly, accessing TOR will save the source MAC address, FLOW-ID, VLAN-TAG and IP 5-tuple to the local table, shown in Table 1. Then TOR perform the normal routing.

Src IP	Dst IP	Src Port	Dst Port	Proto	MAC SA	Flow ID	VLAN TAG
192.168.2.100	192.168.3.30	5678	21	6	0x01a4f5aefe	0xa878	100
10.1.10.2	10.2.20.1	8957	21	6	0xfd16783acd	0xc9a0	1024
192.168.2.100	10.3.50.1	2345	80	6	0x0a25364101	0x0ac9	3
200.1.2.3	100.2.3.4	2567	47	17	0xed16d8ea0a	0x37a0	90

Table 1. FLOW-ID Mapping Table

2)Shown in Figure 3. Congestion caused by Incast flow, T4 detect the congestion in a certain queue and exceed the threshold. Distinguish the flow model according to the priority of the queue.

Please view in a fixed-width font such as Courier.

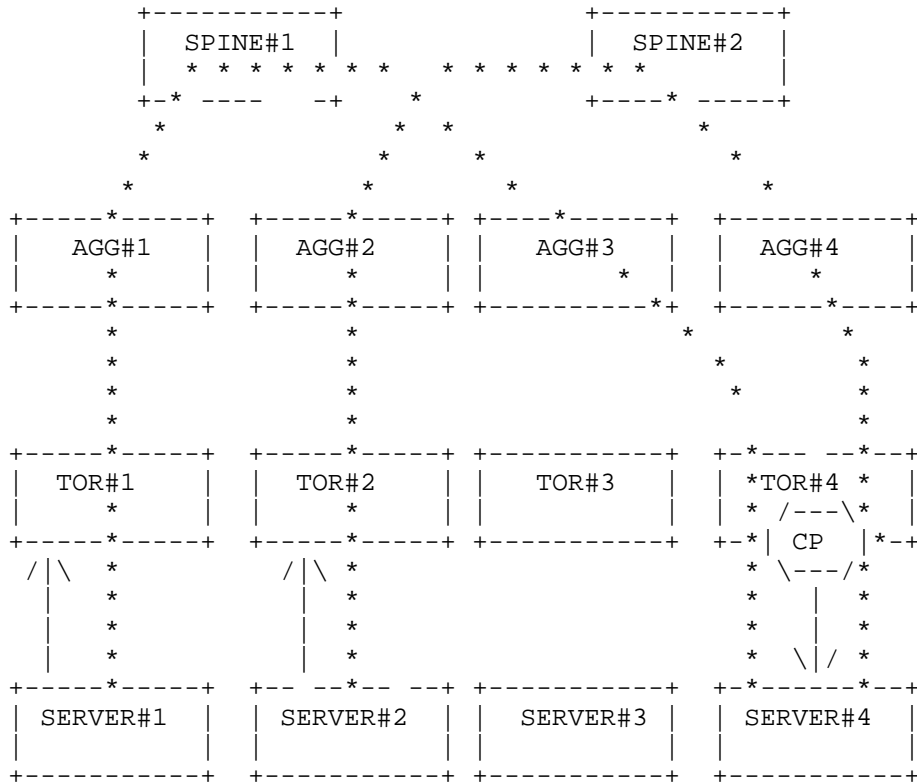


Figure 3. Incast flow model

3) If it is the flow from QCN server, conduct self-defined CNM which include the 5-tuple, congestion indications (defined in QCN specification, such as QntzFb, CPID, Qoffset, QDelta), encapsulate IP +UDP. UDP need to use a specific port No. which is used to recognize the QCN frame in TOR. Or use a bit in the IP head(reserved bit) to indicate the type of the frame. The dedicate IP is set to the source IP which assure the CNM could be routed to the accessing TOR. It's better to construct the self-defined CNM based on the standard CNM to reduce the writing times which might increase the performance.

4) As shown in the Figure 4&5, T2 recognize the self-defined CNM according to the destination UDP port. T2 map the self-defined CNM to the standard CNM and send to H2. The QCN is performed in L2 domain because the adaptor of H2 support the standard QCN.

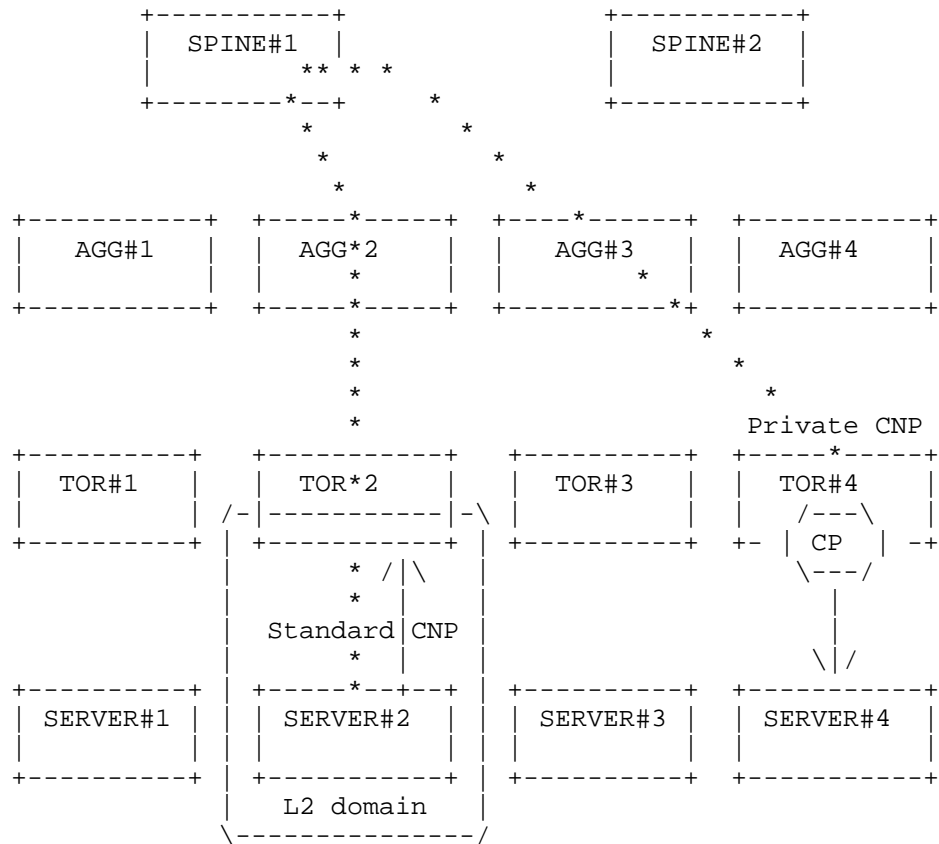
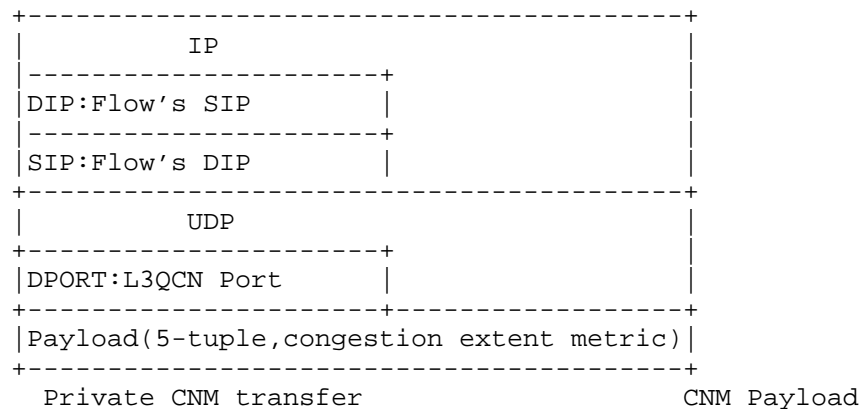


Figure 4. Construct the self-defined CNM

Please view in a fixed-width font such as Courier.



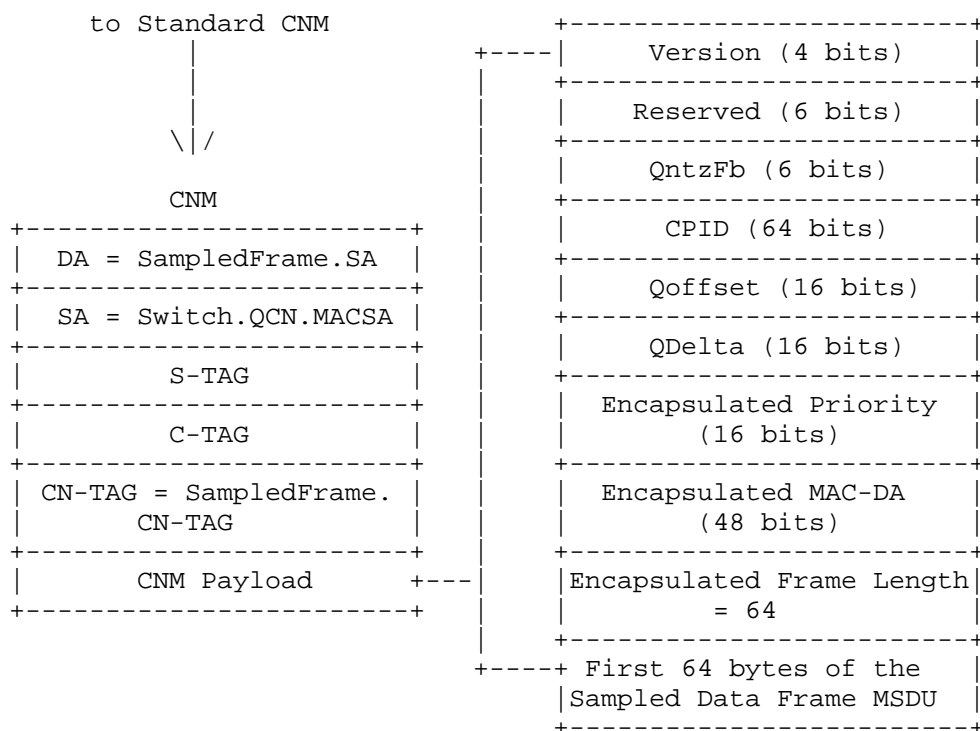
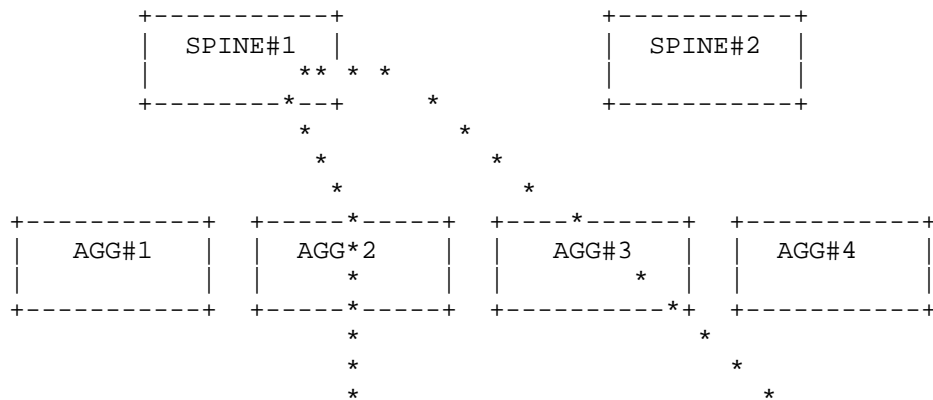


Figure 5. Transfer Private CNM to Standard CNM

5)As shown in the Figure 6 , T4 recognize which flow causes the congestion. CP construct the standard CNP. The adaptor of RoCEv2 server support CNP and reduce the transmission rate according to the probability of CNP reception.

Please view in a fixed-width font such as Courier.



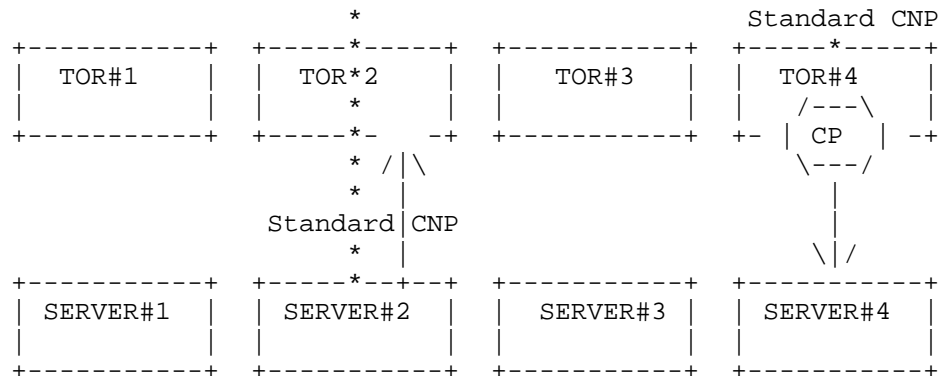


Figure 6. CP construct the standard CNP based on RoCEv2

4. Conclusion

L3QCN resolve the problem that QCN could not support L3 network. L3QCN realize the QCN mechanism across the L3 network. There is no modification on the QCN servers. For the RoCEv2 traffic, since the CP send the CNP when reach the congestion threshold, it reduce the Control Loop Delay dramatically which could reduce the depth of the queue buffer and the datagram delay. The performance of the network is improved.

5 Security Considerations

N/A

6 IANA Considerations

Will apply the specific UDP port No. if required.

7 References

7.1 Normative References

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

7.2 Informative References

- [1] IEEE 802.1: 802.1Qaz Draft 2.5- Enhanced Transmission Selection
- [2] IEEE 802.1: 802.1Qbb Draft 2.3- Priority-based Flow Control
- [3] IEEE 802.1: 802.1Qau Draft 2.4- Congestion Notification

[4] Yibo Zhu et al., SIGCOMM 2015, Congestion Control for Large-Scale RDMA Deployments

[5] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN). RFC 3168

Authors' Addresses

Yolanda Yu
101 SOFTWARE AV., YUHUATAI DIST., NANJING,
JIANGSU, 210012, CHINA
EMail: yolanda.yu@huawei.com