

Transport Area Working Group
Internet-Draft
Obsoletes: 5405 (if approved)
Intended status: Best Current Practice
Expires: December 19, 2014

L. Eggert
NetApp
G. Fairhurst
University of Aberdeen
G. Shepherd
Cisco Systems
June 17, 2014

UDP Usage Guidelines
draft-eggert-tsvwg-rfc5405bis-01

Abstract

The User Datagram Protocol (UDP) provides a minimal message-passing transport that has no inherent congestion control mechanisms. 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.

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, and middlebox traversal.

If published as an RFC, this document will obsolete RFC5405.

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 19, 2014.

Copyright Notice

Copyright (c) 2014 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 not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

Table of Contents

1. Introduction	2
2. Terminology	4
3. UDP Usage Guidelines	4
3.1. Congestion Control Guidelines	5
3.2. Message Size Guidelines	12
3.3. Reliability Guidelines	14
3.4. Checksum Guidelines	15
3.5. Middlebox Traversal Guidelines	17
4. Multicast UDP Usage Guidelines	19
4.1. Multicast Congestion Control Guidelines	20
4.2. Message Size Guidelines for Multicast	22
5. Programming Guidelines	22
5.1. Using UDP Ports	23
5.2. ICMP Guidelines	25
6. Security Considerations	26
7. Summary	27
8. IANA Considerations	29
9. Acknowledgments	29
10. References	29
10.1. Normative References	29
10.2. Informative References	30
Appendix A. Revision Notes	36

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 UDP

(both 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 tear-down 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 link interface, which is often much greater than the available 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". 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, and middlebox traversal.

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 and broadcast, and applications that use UDP tunnels to support IP flows.

Finally, although this document specifically refers to applications that use 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 or control applications that choose to run directly over IP by registering their own IP protocol number with IANA. This document may 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. 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].

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 [RFC0896] 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 3.1.1 discusses congestion control options for applications that perform bulk transfers over UDP. Such applications can employ schemes that sample the path over several subsequent RTTs during which data is exchanged, in order 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 3.1.2 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 3.1.6 discusses congestion control considerations when UDP is used as a tunneling protocol. Section 4 provides additional recommendations for broadcast and multicast usage.

UDP applications may take advantage of Explicit Congestion Notification (ECN), providing that the application programming interface can support ECN and the congestion control can appropriately react to ECN-marked packets. [RFC6679] provides guidance on how to use ECN for UDP-based applications using the Real-Time Protocol (RTP).

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. 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 RTT, SHOULD implement TCP-Friendly Rate Control (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 3.1.1, 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 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.

Finally, some bulk transfer applications may choose not to implement any congestion control mechanism and instead rely on transmitting across reserved path capacity. 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.

3.1.2. 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 3.1.1, 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 round-trip time (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. Applications SHOULD implement the algorithm specified in [RFC6298] to compute a smoothed RTT (SRTT) estimate. They SHOULD also detect packet loss and exponentially back their retransmission timer off when a loss event occurs. When implementing this scheme, applications need to choose a sensible initial value for the RTT. This value SHOULD generally be as conservative as possible for the given application. TCP uses an initial value of 3 seconds [RFC6298], which is also RECOMMENDED as an initial value for UDP applications. SIP [RFC3261] and GIST [RFC5971] use an initial value of 500 ms, and initial timeouts that are shorter than this are likely problematic in many cases. It is also important to note that the initial timeout is not the maximum possible timeout -- the RECOMMENDED algorithm in [RFC6298] yields timeout values after a series of losses that are much longer than the initial value.

Some applications cannot maintain a reliable RTT estimate for a destination. The first case is that of applications that exchange too few UDP datagrams with a peer to establish a statistically accurate RTT estimate. Such applications MAY use a predetermined transmission interval that is exponentially backed-off when packets are lost. TCP uses an initial value of 3 seconds [RFC6298], which is also RECOMMENDED as an initial value for UDP applications. 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.

A second class 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. The 3-second interval was chosen based on TCP's retransmission timeout when the RTT is unknown [RFC6298], and shorter values are likely problematic in many cases. Note that the sending rate in this case must be more conservative than in the two 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.

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.

3.1.3. 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 has also been shown to improve the coexistence of TCP flows with other flows.

Even low data-volume UDP flows may benefit from rate control, 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.

3.1.4. QoS, Pre-Provisioned or Reserved Capacity

An application using UDP can use the differentiated services and integrated services QoS frameworks. These are usually 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.

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. For this reason, and to protect other applications sharing the same path, applications SHOULD deploy an appropriate circuit breaker, as described in Section 3.1.5. 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 used in networks within a controlled environment may be able to exploit network management functions to detect whether they are causing congestion, and react accordingly.

3.1.5. 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.fairhurst-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. Later sections provide examples of cases where circuit breakers may or may not be desirable.

[I-D.fairhurst-tsvwg-circuit-breaker] provides guidance on the use of circuit breakers and examples of usage. The use of a circuit breaker in RTP is specified in [I-D.ietf-avtcore-rtp-circuit-breakers].

3.1.6. UDP Tunnels

One increasingly popular use of UDP is as a tunneling protocol, 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. 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 traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. Consequently, a tunnel carrying IP-based traffic should already interact appropriately with other traffic sharing the path, and specific congestion control mechanisms for the tunnel are not necessary.

However, if the IP traffic in the tunnel is known to not be congestion-controlled, additional measures are RECOMMENDED in order to limit the impact of the tunneled traffic on other traffic sharing the path.

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.

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. Because tunnels are usually bulk-transfer applications as far as the intermediate routers are concerned, the guidelines in Section 3.1.1 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.

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, e.g., handling fragmentation, generating and responding to ICMP messages, etc. 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 MTU of 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 itself [RFC1191][RFC1981][RFC4821] to determine whether the path to a destination will support its desired message size without fragmentation.

Applications that do not follow this recommendation to do PMTU 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 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, 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.

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. 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. PLPMTUD therefore places additional design requirements on a UDP application that wishes to use this method.

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. 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** verify that their application handles 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.

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.

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) 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 are provided in Section 3.1 of [RFC6396].

Applications SHOULD enable UDP checksums. For IPv4, [RFC0768] permits the option to disable their use. The use of the UDP checksum was required when applications transmit UDP over IPv6 [RFC2460]. This requirement was updated in [RFC6395], but only for specific protocols and applications, and the implementation of the set of functions defined in [RFC6396] is then REQUIRED. These additional design requirements for using a zero IPv6 UDP checksum [RFC6396] are not present for IPv4, since the network-layer header validates information that is not protected for an IPv6 packet.

Applications that choose to disable UDP checksums when transmitting over IPv4 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. 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 a checksum coverage length value: 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.

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).

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 and endpoints.

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 may 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 period of 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. 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-alives 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 network 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 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.

Sending keep-alives is not a substitute for implementing a mechanism to recover from broken sessions. Like all UDP datagrams, keep-alives 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-alives. 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-alives 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-alives 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 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 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). This has implications on applications that use multiple UDP flows in parallel, even on multiple ports Section 5.1.1.

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 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]. Anycast senders 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. These MUST follow

the same requirements as a tunnel carrying unicast data (see Section 3.1.6). 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 bandwidth, 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 mechanism 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.

Congestion control mechanisms for multicast may operate on longer timescales than for unicast (e.g., due to the higher group RTT of a heterogeneous group); appropriate methods are particularly for any multicast session where all or part of the multicast distribution tree spans an access network (e.g., a home gateway).

Multicast congestion control needs to consider 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. 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. A multicast congestion control method MAY therefore decide not to reduce the rate of the entire multicast group in response to a report received by a single receiver; instead it can decide to expel each congested receiver from the multicast group and then distribute content to these congested receivers at a lower-rate using unicast congestion-control. Care needs to be taken when

this action results in many flows being simultaneously transitioned, so that this does not result in excessive traffic exasperating congestion and potentially contributing to congestion collapse.

Some classes of multicast applications support real-time transmissions in which the quality of the transfer may be monitored at the receiver. Applications that detect a significant reduction in user quality SHOULD regard this as a congestion signal (e.g., to leave a group using layered multicast encoding).

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 3.1.2 are also applicable to such 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.

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.

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].

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 on 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 in this case. 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 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 [I-D.ietf-tsvwg-port-use].

A UDP sender SHOULD NOT use a zero source port value, and a UDP receiver should not bind to port zero. Applications SHOULD implement corresponding receiver 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). Although the source port value is often not directly used in multicast applications, this should still be set to a random or pre-determined value.

The UDP port number fields have been used as a basis to design load-balancing solutions for IPv4. This approach has also been leveraged for IPv6 [RFC6438], but the IPv6 "flow label" [RFC6437] may also be used as a basis for entropy for load balancing. This use of the flow label for load balancing is consistent with the intended use, 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 were specified. Router vendors are encouraged to start using the 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. 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 5-tuples). [RFC6335] recommends applications 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 impact 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 can not 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 congestion control to react to congestion events, unless the congestion state is shared between all flows in a session.

Therefore, applications **MUST NOT** assume consistent behavior of middleboxes when multiple UDP flows are used; many devices respond differently as the number of ports used 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 3.1.4.

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, e.g., transient ICMP "unreachable" messages should not normally cause a communication abort.

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. Applications that respond to short requests with potentially large responses are vulnerable to amplification attacks, and SHOULD authenticate the sender before responding. The source IP address of a request is not a useful authenticator, because it can easily be spoofed.

One option of 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) [RFC5996] 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.

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 of securing UDP communications is through Datagram Transport Layer Security (DTLS) [RFC6347]. 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. The IETF standard for securing RTP [RFC3550] communication sessions over UDP is the Secure Real-time Transport Protocol (SRTP) [RFC3711]. 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 (especially when the intended endpoint is on the same link as the sender). This is a lightweight mechanism that 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.fairhurst-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.

7. Summary

This section summarizes the guidelines made in Sections 3 and 6 in a tabular format (Table 1) for easy referencing.

Recommendation	Section
MUST tolerate a wide range of Internet path conditions	3
SHOULD use a full-featured transport (TCP, SCTP, DCCP)	
SHOULD control rate of transmission	3.1
SHOULD perform congestion control over all traffic	
for bulk transfers,	3.1.1
SHOULD consider implementing TFRC	
else, SHOULD in other ways use bandwidth similar to TCP	
for non-bulk transfers,	3.1.2
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	
for tunnels carrying IP Traffic,	3.1.6
SHOULD NOT perform congestion control	
for non-IP tunnels or rate not determined by traffic,	3.1.6
SHOULD perform congestion control	
SHOULD NOT send datagrams that exceed the PMTU, i.e.,	3.2
SHOULD discover PMTU or send datagrams < minimum PMTU; Specific application mechanisms are REQUIRED if PLPMTUD is used.	
SHOULD handle datagram loss, duplication, reordering	3.3
SHOULD be robust to delivery delays up to 2 minutes	
SHOULD enable IPv4 UDP checksum	3.4
SHOULD enable IPv6 UDP checksum; Specific application mechanisms are REQUIRED if a zero IPv6 UDP checksum is used.	
else, MAY use UDP-Lite with suitable checksum coverage	3.4.1
SHOULD NOT always send middlebox keep-alives	3.5
MAY use keep-alives when needed (min. interval 15 sec)	
MUST check IP source address	5

and, for client/server applications	
SHOULD send responses from src address matching request	
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.

10. References

10.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.
- [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.
- [RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, June 2011.

10.2. Informative References

- [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.fairhurst-tsvwg-circuit-breaker]
Fairhurst, G., "Network Transport Circuit Breakers", draft-fairhurst-tsvwg-circuit-breaker-01 (work in progress), May 2014.
- [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-avtcore-rtp-circuit-breakers]
Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", draft-ietf-avtcore-rtp-circuit-breakers-05 (work in progress), February 2014.
- [I-D.ietf-tsvwg-port-use]
Touch, J., "Recommendations for Transport Port Uses", draft-ietf-tsvwg-port-use-04 (work in progress), May 2014.
- [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.
- [RFC0896] Nagle, J., "Congestion control in IP/TCP internetworks", RFC 896, January 1984.
- [RFC0919] Mogul, J., "Broadcasting Internet Datagrams", STD 5, RFC 919, October 1984.
- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.
- [RFC1536] Kumar, A., Postel, J., Neuman, C., Danzig, P., and S. Miller, "Common DNS Implementation Errors and Suggested Fixes", RFC 1536, October 1993.
- [RFC1546] Partridge, C., Mendez, T., and W. Milliken, "Host Anycasting Service", RFC 1546, November 1993.
- [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, April 1998.
- [RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, August 1999.
- [RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.
- [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, August 2000.

- [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, January 2001.
- [RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, June 2001.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC3738] Luby, M. and V. Goyal, "Wave and Equation Based Rate Control (WEBRC) Building Block", RFC 3738, April 2004.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [RFC3819] Karn, P., 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, July 2004.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, March 2006.
- [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, March 2006.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.
- [RFC4654] Widmer, J. and M. Handley, "TCP-Friendly Multicast Congestion Control (TFMCC): Protocol Specification", RFC 4654, August 2006.
- [RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, November 2007.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.
- [RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007.
- [RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, October 2007.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

- [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, August 2009.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.
- [RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, November 2009.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.
- [RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, April 2010.
- [RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", RFC 5971, October 2010.
- [RFC5973] Stiemerling, M., Tschofenig, H., Aoun, C., and E. Davies, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", RFC 5973, October 2010.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.
- [RFC6395] Gulrajani, S. and S. Venaas, "An Interface Identifier (ID) Hello Option for PIM", RFC 6395, October 2011.
- [RFC6396] Blunk, L., Karir, M., and C. Labovitz, "Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format", RFC 6396, October 2011.

- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, November 2011.
- [RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, November 2011.
- [RFC6513] Rosen, E. and R. Aggarwal, "Multicast in MPLS/BGP IP VPNs", RFC 6513, February 2012.
- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.
- [RFC6726] Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport", RFC 6726, November 2012.
- [RFC6807] Farinacci, D., Shepherd, G., Venaas, S., and Y. Cai, "Population Count Extensions to Protocol Independent Multicast (PIM)", RFC 6807, December 2012.
- [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. Revision Notes

Note to RFC-Editor: please remove this entire section prior to publication.

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

TSVWG
Internet-Draft
Intended status: Informational
Expires: May 16, 2015

R. Geib, Ed.
Deutsche Telekom
D. Black
EMC Corporation
November 12, 2014

DiffServ interconnection classes and practice
draft-geib-tsvwg-diffserv-intercon-08

Abstract

This document proposes a limited and well defined set of DiffServ PHBs and codepoints to be applied at (inter)connections of two separately administered and operated networks. Many network providers operate MPLS using Treatment Aggregates for traffic marked with different DiffServ PHBs, 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.

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 16, 2015.

Copyright Notice

Copyright (c) 2014 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
2. MPLS and the Short Pipe tunnel model	5
3. An Interconnection class and codepoint scheme	6
3.1. End-to-end QoS: PHB and DS CodePoint Transparency	11
3.2. Treatment of Network Control traffic at carrier interconnection interfaces	12
4. Acknowledgements	13
5. IANA Considerations	13
6. Security Considerations	13
7. References	13
7.1. Normative References	13
7.2. Informative References	14
Appendix A. Annex A Carrier interconnection related DiffServ aspects	15
Appendix B. Annex 2 The MPLS Short Pipe Model and IP traffic . .	17
Appendix C. Change log	21
Authors' Addresses	21

1. Introduction

DiffServ has been deployed in many networks. As described by section 2.3.4.2 of RFC 2475, remarking of packets at domain boundaries is a DiffServ feature [RFC2475]. This draft proposes a set of standard QoS classes and code points at interconnection points to which and from which locally used classes and code points should be mapped.

RFC2474 specifies the DiffServ Codepoint Field [RFC2474]. Differentiated treatment is based on the specific DSCP. Once set, it may change. 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. Many networks do not follow this recommendation, and instead remark unknown or unexpected DSCPs to the zero DSCP for consistency with default (best effort) forwarding.

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 meet its Service Level Agreements (SLAs). Based on that foundation, foundation, [RFC5127] introduces the concept of DiffServ Treatment

Aggregates, which enable traffic marked with multiple DSCPs to be forwarded in a single MPLS Traffic Class (TC). Like RFC 5127, this document assumes robust provider backbone traffic engineering.

RFC5127 recommends transmission of DSCPs as they are received. This is not possible, if the receiving and the transmitting domains at a network interconnection use different DSCPs for the PHBs involved.

This document is motivated by requirements for IP network interconnection with DiffServ support among providers that operate MPLS in their backbones, but is applicable to other technologies. The operational simplifications and methods in this document help align IP DiffServ functionality with MPLS limitations, particularly when MPLS penultimate hop popping is used. That is an important reason why this document specifies 4 interconnection Treatment Aggregates. Limiting DiffServ to a small number Treatment Aggregates can help ensure that network traffic leaves a network with the same DSCPs that it was received with. The approach proposed here may be extended by operators or future specifications.

In isolation, use of standard interconnection PHBs and DSCPs may appear to be additional effort for a network operator. The primary offsetting benefit is that the 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. Otherwise, the PHBs and DSCPs have to be negotiated and configured independently for each network interconnection, which has poor scaling properties. Further, end-to-end QoS 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. This document supports one-to-one DSCP remarking at network interconnections (not n DSCP to one DSCP remarking).

The example given in RFC 5127 on aggregation of DiffServ service classes uses 4 Treatment Aggregates, and this document does likewise because:

- o The available coding space for carrying QoS information (e.g., DiffServ PHB) in MPLS and Ethernet is only 3 bits in size, and is intended for more than just QoS purposes (see e.g. [RFC5129]).
- o There should be unused codes for interconnection purposes. This leaves space for future standards, for private bilateral agreements and for local use PHBs and DSCPs.
- o Migrations from one code point scheme to another may require spare QoS code points.

RFC5127 provides recommendations on aggregation of DSCP-marked traffic into MPLS Treatment Aggregates and offers a deployment example [RFC5127] that does not work for the MPLS Short Pipe model when that model is used for ordinary network traffic. This document supports the MPLS Short Pipe model for ordinary network traffic and hence differs from the RFC5127 approach as follows:

- o remarking of received DSCPs to domain internal DSCPs is to be expected for ordinary IP traffic at provider edges (and for outer headers of tunneled IP traffic).
- o document follows RFC4594 in the proposed marking of provider Network Control traffic and expands RFC4594 on treatment of CS6 marked traffic at interconnection points (see section 3.2).

This document is organized as follows: section 2 reviews the MPLS Short Pipe tunnel model for DiffServ Tunnels [RFC3270]; effective support for that model is a crucial goal of this document. Section 3 introduces DiffServ interconnection Treatment Aggregates, plus the PHBs and DSCPs that are mapped to these Treatment Aggregates. Further, section 3 discusses treatment of non-tunneled and tunneled IP traffic and MPLS VPN QoS aspects. Finally Network Management PHB treatment is described. Annex A discusses how domain internal IP layer QoS schemes impact interconnection. Annex B describes the impact of the MPLS Short Pipe model (pen ultimate hop popping) on QoS related IP interconnections.

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. RFC 5160 suggests Meta-QoS-Classes to enable deployment of standardized end to end QoS classes [RFC5160]. In private discussion, the authors of that RFC agree that the proposed interconnection class- and codepoint scheme and its enablement of standardised end to end classes would complement their own work.

Work on signaling Class of Service at interconnection interfaces by BGP [I-D.knoll-idr-cos-interconnect], [ID.idr-sla] is beyond the scope of this draft. When the basic DiffServ elements for network interconnection are used as described in this document, signaled access to QoS classes may be of interest. These two BGP documents focus on exchanging SLA and traffic conditioning parameters and assume that common PHBs identified by the signaled DSCPs have been established prior to BGP signaling of QoS.

2. MPLS and the Short Pipe tunnel model

The Pipe and Uniform models for Differentiated Services and Tunnels are defined in [RFC2983]. RFC3270 adds the MPLS Short Pipe model in order to support penultimate hop popping (PHP) of MPLS Labels, primarily for IP tunnels and VPNs. The Short Pipe model and PHP have become popular with many network providers that operate MPLS networks and are now widely used for ordinary network traffic, not just traffic encapsulated in IP tunnels and VPNs. This has important implications for DiffServ functionality in MPLS networks.

RFC 2474's recommendation to forward traffic with unrecognized DSCPs with Default (best effort) service without rewriting the DSCP has proven to be a poor operational 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, so a common practice is to use CS0 to remark 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 TC field and allow that to differ from the PHB indicated by the DSCP in the MPLS-encapsulated IP packet. That would allow an unrecognized DSCP to be carried edge-to-edge over an MPLS network, because the effective DSCP used by the MPLS network would be encoded in the MPLS label TC field (and also carried edge-to-edge); this approach assumes that a provider MPLS label with the provider's TC field being present at all hops within the provider's network.

The Short Pipe tunnel model and PHP violate that assumption because PHP pops and discards the MPLS provider label carrying the provider's TC field. That discard occurs one hop upstream of the MPLS tunnel endpoint, resulting in no provider TC info being available at tunnel egress. Therefore the DSCP field in the MPLS-encapsulated IP header has to contain a DSCP that is valid for the provider's network; propagating another DSCP edge-to-edge requires an IP tunnel of some form. In the absence of IP tunneling (a common case for MPLS networks), it is not possible to pass all 64 possible DSCP values edge-to-edge across an MPLS network. See Annex B for a more detailed discussion.

If transport of a large number (much greater than 4) DSCPs is required across a network that supports this DiffServ interconnection scheme, a tunnel or VPN can be provisioned for this purpose, so that the inner IP header carries the DSCP that is to be preserved not to be changed. From a network operations perspective, the customer

equipment (CE) is the preferred location for tunnel termination, although a receiving domains Provider Edge router is another viable option.

3. An Interconnection class and codepoint scheme

At an interconnection, the networks involved need to agree on the PHBs used for interconnection and the specific DSCP for each PHB. This may involve remarking for the interconnection; such remarking is part of the DiffServ Architecture [RFC2475], at least for the network edge nodes involved in interconnection. See Annex A for a more detailed discussion. This draft proposes a standard interconnection set of 4 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 her internal scheme. The set of DSCPs and PHBs supported across the two interconnected domains and the treatment of PHBs and DSCPs not recognized by the receiving domain should be part of the interconnect SLA.

RFC 5127's four treatment aggregates include a Network Control aggregate for routing protocols and OAM traffic that is essential for network operation administration, control and management. Using this aggregate as one of the four in RFC 5127 implicitly assumes that network control traffic is forwarded in potential competition with all other network traffic, and hence DiffServ must favor such traffic (e.g., via use of the CS6 codepoint) for network stability. That is a reasonable assumption for IP-based networks where routing and OAM protocols are mixed with all other types of network traffic; corporate networks are an example.

In contrast, mixing of all traffic is not a reasonable assumption for MPLS-based provider or carrier networks, where customer traffic is usually segregated from network control (routing and OAM) traffic via other means, e.g., network control traffic use of separate LSPs that can be prioritized over customer LSPs (e.g., for VPN service) via other means. This sort of network control traffic from customer traffic is also used for MPLS-based network interconnections. In addition, many customers of a network provider do not exchange Network Control traffic (e.g., routing) with the network provider. For these reasons, a separate Network Control traffic aggregate is not important for MPLS-based carrier or provider networks; when such traffic is not segregated from other traffic, it may reasonably share the Assured Elastic treatment aggregate (as RFC 5127 suggests for a situation in which only three treatment aggregates are supported).

In contrast, VoIP is emerging as a valuable and important class of network traffic for which network-provided QoS is crucial, as even

minor glitches are immediately apparent to the humans involved in the conversation.

For these reasons, the Diffserv Interconnect scheme in this document departs from the approach in RFC 5127 by not providing a Network Control traffic aggregate, and instead dedicating the fourth traffic aggregate for VoIP traffic. Network Control traffic may still be exchanged across network interconnections, see Section 3.2 for further discussion.

Similar approaches to use of 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], GSM IR.34 [IR.34] and MEF23.1 [MEF23.1].

The list of the four Diffserv Interconnect traffic aggregates follows, highlighting differences from RFC 5127 and the specific traffic classes from RFC 4594 that each class aggregates.

Telephony Service Treatment Aggregate: PHB EF, DSCP 101 110 and VOICE-ADMIT, DSCP 101100, see [RFC3246] , [RFC4594][RFC5865]. This Treatment Aggregate corresponds to RFC 5127's real time Treatment Aggregate definition regarding the queuing, but it is restricted to transport Telephony Service Class traffic in the sense of RFC 4594.

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 designed to transport the portions of RFC 5127's Real Time Treatment Aggregate, which consume large amounts of bandwidth, namely Broadcast Video, Real-Time Interactive and Multimedia Conferencing. The treatment aggregate should be configured with a rate queue (which is in line with RFC 4594 for the mentioned traffic classes). As compared to RFC 5127, the number of DSCPs has been reduced to one (initially) and the proposed queuing mechanism. The latter is however in line with RFC4594.

Assured Elastic Treatment Aggregate This Treatment Aggregate consists of the entire AF3 PHB group AF3, i.e., DSCPs 011 010, 011 100 and 011 110. As compared to RFC5127, just the number of DSCPs, which has been reduced. This document suggests to transport signaling marked by AF31. RFC5127 suggests to map Network Management traffic into this Treatment Aggregate, if no separate Network Control Treatment Aggregate is supported (for a more detailed discussion of

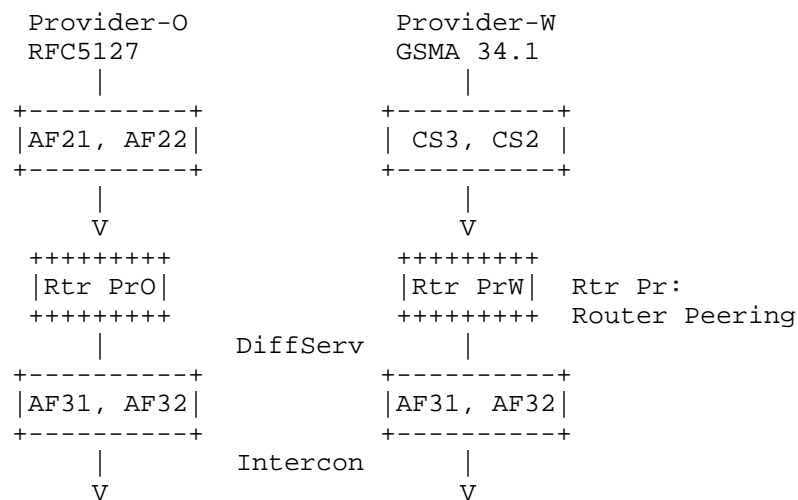
Network Control PHB treatment see section 3.2). GSMA IR.34 proposes to transport signaling traffic by AF31 too.

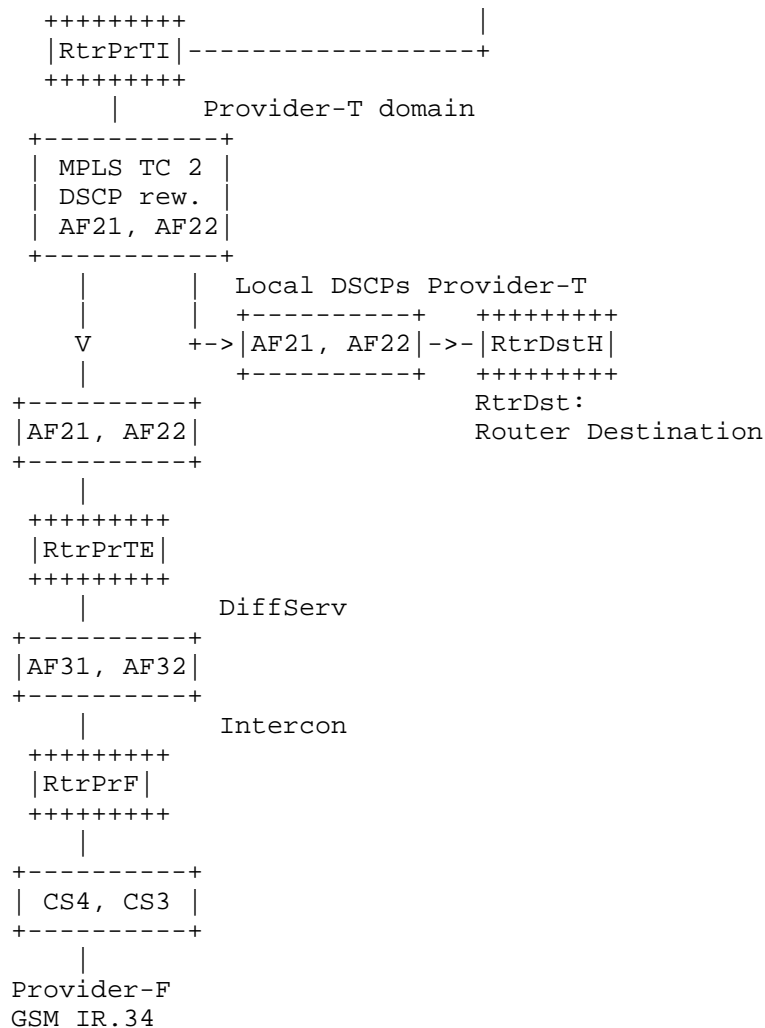
Default / Elastic Treatment Aggregate: transports the default PHB, CS0 with DSCP 000 000. RFC 5127 example refers to this Treatment Aggregate as Aggregate Elastic. An important difference as compared to RFC5127 is that any traffic with unrecognized or unsupported DSCPs may be remarked to this DSCP.

RFC 4594's Multimedia Streaming class has not been mapped to the above scheme. By the time of writing, the most popular streaming applications use TCP transport and adapt picture quality in the case of congestion. These applications are proprietary and still change behaviour frequently. At this state, the Bulk Real-Time Treatment Aggregate or the Bulk Real-Time Treatment Aggregate may be a reasonable match.

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 QoS interconnection SLA.

Traffic of provider O terminates within provider Ts network, while provider W's traffic transits through the network of provider T to provider F. Assume all providers to run their own internal codepoint schemes for a PHB group with properties of the DiffServ Intercon Assured Treatment Aggregate.





DiffServ Intercon example

Figure 1

It is easily visible that all providers only need to deploy internal DSCP to DiffServ Intercon DSCP mappings to exchange traffic in the desired classes. Provider W has decided that the properties of his internal classes CS3 and CS2 are best met by the Diffserv Intercon Assured Elastic Treatment Aggregate, PHBs AF31 and AF32 respectively. At the outgoing peering interface connecting provider W with provider

T remarks CS3 traffic to AF31 and CS 2 traffic to CS32. The domain internal PHBs of provider T meeting the Diffserv Intercon Assured Elastic Treatment Aggregate requirements is AF2. Hence AF31 traffic received at the interconnection with provider T is remarked to AF21 by the peering router of domain T. As domain T deploys MPLS, further the MPLS TC is set to 2. Traffic received with AF32 is remarked to AF22. The MPLS TC of the Treatment Aggregate is the same, TC 2. At the pen-ultimate MPLS node, the top MPLS label is removed. The packet should be forwarded as determined by the incoming MPLS TC. The peering router connecting domain T with domain F classifies the packet by its domain T internal DSCP AF21 for the Diffserv Intercon Assured Elastic Treatment Aggregate. As it leaves domain T on the interface to domain F, it is remarked to AF31. The peering router of domain F classifies the packet for domain F internal PHB CS4, as this is the PHB with properties matching Diffserv Intercon's Assured Elastic Treatment Aggregate. Likewise, AF21 traffic is remarked to AF32 by the peering router of domain T when leaving it and from AF32 to CS3 by domain F's peering router when receiving it.

This example can be extended. Suppose Provider-O also supports a PHB marked by CS2 and this PHB is supposed to be transported by QoS within Provider-T domain. Then Provider-O will remark it with a DSCP other than AF31 DSCP in order to preserve the differentiation from CS2; AF11 is one possibility that might be private to the interconnection between Provider-O and Provider-T; there's no assumption that Provider-W can also use AF11, as it may not be in the SLA with Provider-W.

Now suppose Provider-W supports CS2 for internal use only. Then no Diffserv intercon DSCP mapping may be configured at the peering router. Traffic, sent by Provider-W to Provider-T marked by CS2 due to a misconfiguration may be remarked to CS0 by Provider-T.

See section 3.1 for further discussion of this and DSCP transparency in general.

RFC5127 specifies a separate Treatment Aggregate for network control traffic. It may be present at interconnection interfaces too, but depending on the agreement between providers, Network Control traffic may also be classified into a different interconnection class. See section 3.2 for a detailed discussion on the treatment of Network Control traffic.

RFC2575 states that Ingress nodes must condition all other 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 the Default PHB codepoint. 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.

The proposed interconnect class and code point scheme is designed for point to point IP layer interconnections among MPLS networks. Other types of interconnections are out of scope of this document. The basic class and code point scheme is applicable on Ethernet layer too, if a provider e.g. supports Ethernet priorities like specified by IEEE 802.1p.

3.1. End-to-end QoS: PHB and DS CodePoint Transparency

This section describes how the use of a common PHB and DSCP scheme for interconnection can lead to end-to-end DiffServ-based QoS across networks that do not have common policies or practices for PHB and DSCP usage. This will initially be possible for PHBs and DSCPs corresponding to at most 3 or 4 Treatment Aggregates due to the MPLS considerations discussed previously.

Networks can be expected to differ in the number of PHBs available at interconnections (for terminating or transit service) and the DSCP values used within their domain. At an interconnection, Treatment Aggregate and PHB properties are best described by SLAs and related explanatory material. See annex A for a more detailed discussion about why PHB and DSCP usage is likely to differ among networks. For the above reasons and the desire to support interconnection among networks with different DiffServ schemes, the DiffServ interconnection scheme supports a small number of PHBs and DSCPs; this scheme is expandable.

The basic idea is that traffic sent with a DiffServ interconnect PHB and DSCP is restored to that PHB and DSCP (or a PHB and DSCP within the AF3 PHB group for the Assured Treatment Aggregate) at each network interconnection, even though a different PHB and DSCP may be used by each network involved. So, Bulk Inelastic traffic could be sent with AF41, remarked to CS3 by the first network and back to AF41 at the interconnection with the second network, which could mark it to CS5 and back to AF41 at the next interconnection, etc. The result is end-to-end QoS treatment consistent with the Bulk Inelastic Traffic Aggregate, and that is signaled or requested by the AF41 DSCP at each network interconnection in a fashion that allows each network operator to use their own internal PHB and DSCP scheme.

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.

3.2. Treatment of Network Control traffic at carrier interconnection interfaces

As specified by RFC4594, section 3.2, Network Control (NC) traffic marked by CS6 is to be expected at interconnection interfaces. This document does not change NC specifications of 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 RFC 4594. A specific example is that some CS6 traffic exchanged across carrier interconnections is terminated at the network ingress node (e.g., if BGP is running between two routers on opposite ends of an interconnection link), which is consistent with RFC 4594's recommendation to not use CS6 when forwarding CS6-marked traffic originating from user-controlled end points.

The end-to-end QoS discussion in the previous section (3.1) is generally inapplicable to network control traffic - network control traffic is generally intended to control a network, not be transported across it. One exception is that network control traffic makes sense for a purchased transit agreement, and preservation of CS6 for network control traffic that is transited is reasonable in some cases. Use of an IP tunnel is suggested in order to reduce the risk of CS6 markings on transiting network control traffic being interpreted by the network providing the transit.

If the MPLS Short Pipe model is deployed for non tunneled 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 which isn't terminating at the interconnected ingress node:

- o classification of traffic which is network control traffic for both domains. This traffic should be classified and marked for the NC PHB.
- o classification of traffic which is network control traffic for the sending domain only. This traffic should be classified for a PHB

offering similar properties as the NC class (e.g. AF31 as specified by this document). As an example GSMA IR.34 proposes an Interactive class / AF31 to carry SIP and DIAMETER traffic. While this is service control traffic of high importance to the interconnected Mobile Network Operators, it is certainly no Network Control traffic for a fixed network providing transit. The example may not be perfect. It was picked nevertheless because it refers to an existing standard.

- o any other CS6 marked traffic should be remarked or dropped.

4. Acknowledgements

Al Morton and Sebastien Jobert provided feedback on many aspects during private discussions. Mohamed Boucadair and Thomas Knoll helped adding awareness of related work. Fred Baker and Brian Carpenter provided intensive feedback and discussion.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

This document does not introduce new features, it describes how to use existing ones. The security section of RFC 2475 [RFC2475] and RFC 4594 [RFC4594] apply.

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.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.
- [RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.

- [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, March 2002.
- [RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, April 2002.
- [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, May 2002.
- [RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, January 2008.
- [RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, February 2009.
- [RFC5865] Baker, F., Polk, J., and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, May 2010.
- [min_ref] authSurName, authInitials., "Minimal Reference", 2006.

7.2. Informative References

- [I-D.knoll-idr-cos-interconnect] Knoll, T., "BGP Class of Service Interconnection", draft-knoll-idr-cos-interconnect-13 (work in progress), November 2014.
- [ID.idr-sla] IETF, "Inter-domain SLA Exchange", IETF, <http://datatracker.ietf.org/doc/draft-ietf-idr-sla-exchange/>, 2013.
- [IEEE802.1Q] IEEE, "IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks", 2005.
- [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.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000.
- [RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, August 2006.
- [RFC5127] Chan, K., Babiarz, J., and F. Baker, "Aggregation of Diffserv Service Classes", RFC 5127, February 2008.
- [RFC5160] Levis, P. and M. Boucadair, "Considerations of Provider-to-Provider Agreements for Internet-Scale Quality of Service (QoS)", RFC 5160, March 2008.
- [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. Annex A Carrier interconnection related DiffServ aspects

This annex provides a general discussion of PHB and DSCP mapping at IP interconnection interfaces. It also informs about limitations and likely DSCP changes.

The following scenarios start from a domain sending non-tunneled IP traffic using a PHB and a corresponding DSCP to an interconnected domain. The receiving domain may

- o Support the PHB and offer the same corresponding DSCP.
- o Not support the PHB and use the DSCP for a different PHB.
- o Not support the PHB and not use the DSCP.
- o Support the PHB with a differing DSCP, and the DSCP of the sending domain is not used for another PHB
- o Support the PHB with a differing DSCP, and the DSCP of the sending domain is used for another PHB.

RFC2475 allows for local use PHB groups which are only available within a domain. If such a local use PHB is present, non-tunneled IP traffic possibly cannot utilize 64 DSCPs end-to-end.

If a domain receives traffic for a PHB, which it does not support, there are two general scenarios:

- o The received DSCP is not available for usage within the domain.
- o The received DSCP is available for usage within the domain.

RFC2474 suggests to transport packets received with unrecognized DSCPs by the default PHB and leave the DSCP as received. Also if a particular DSCP is spare within a domain, it may later change its QoS design and assign a PHB to a formerly unused DSCP (which a customer used to transit through this unrecognized DSCP will note, as his DSCP will then be remarked). A transparent transport of the same DSCP as unknown with the default PHB may no longer be possible. Remarking to another DSCP apart from the Default PHB's DSCP does not seem to be a good option in the latter case. Which other DSCP is making sense? If a domain interconnects with many other domains, the questions asked here may have to be answered multiple times.

The scenarios above indicate, that reliably delivering a non-tunneled IP packet by the same PHB and DSCP unchanged end-to-end is only likely, if both domains support this DSCP and use the same corresponding DSCP.

Limitations in the number of supported PHBs are to be expected if DiffServ is applied across different domains. Unchanged end-to-end DSCPs should only be expected for non-tunneled IP traffic, if the PHB and DSCP are well specified and generally deployed. This is true for Default Forwarding. EF PHB is a candidate. The Network Control PHB is a local use only example, hence end-to-end support of CS6 for non-tunneled IP traffic at interconnection points should only be expected, if the receiving domain regards this traffic as Network Control traffic relevant for the own domain too.

DiffServ Intercon proposes a well defined set of PHBs and corresponding DSCPs at interconnection points. A PHB to DSCPs correspondence is specified at least for interconnection interfaces. Supported PHBs should be available end-to-end, but domain internal DSCPs may change end-to-end.

Appendix B. Annex 2 The MPLS Short Pipe Model and IP traffic

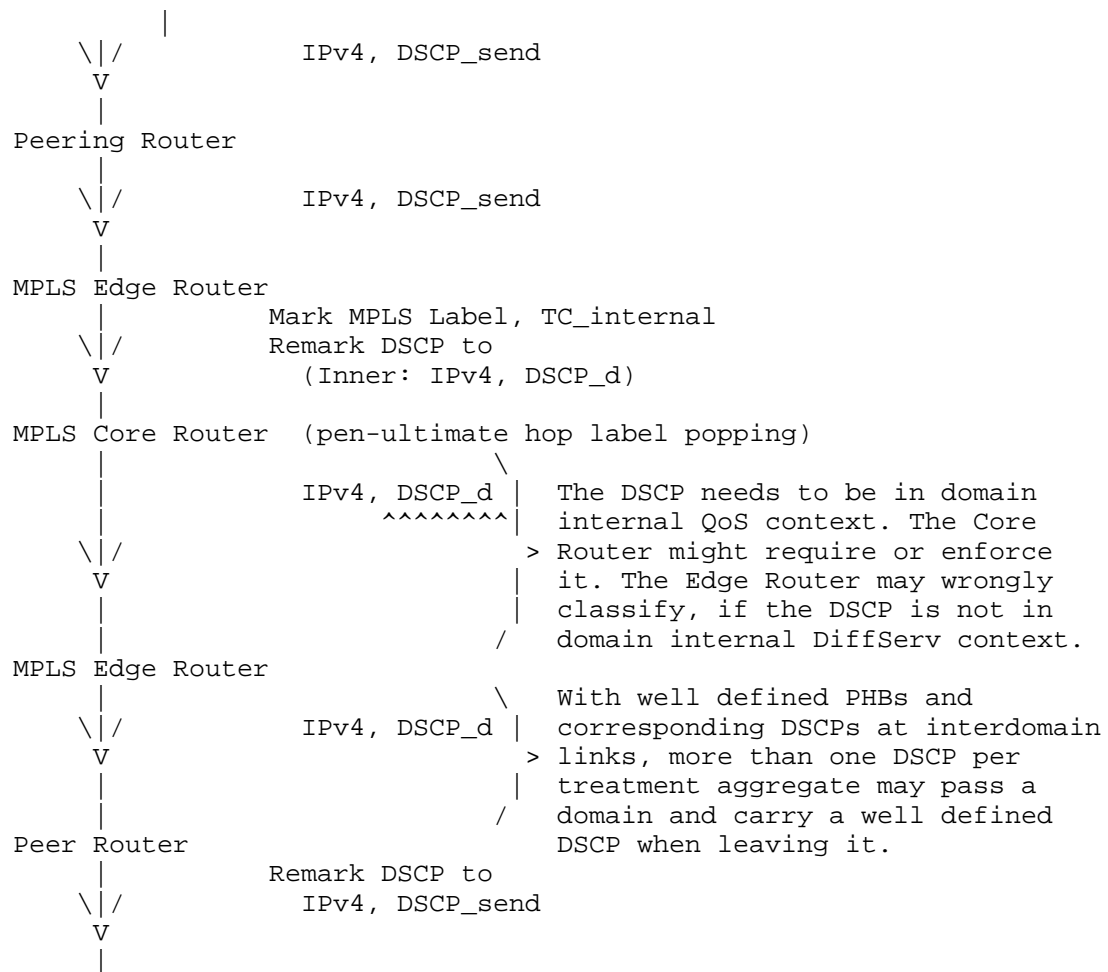
The MPLS Short Pipe Model (or Pen-ultimate Hop Label Popping) is widely deployed by IP carriers. If non-tunneled IPv4 traffic is transported using MPLS Short Pipe, IP headers appear inside the last section of the MPLS domain. This likely impacts the number of PHBs and DSCPs a network provider supports for this kind of traffic. Figure 2 provides an example for the treatment of this kind of traffic.

In the case of tunneled IPv4 traffic, only the outer tunnel header is exposed. Assuming the tunnel not to terminate within the MPLS network section, only the outer tunnel DSCP is impacted.

Non-tunneled IPv6 traffic and Layer 2 and Layer 3 VPN traffic all use an additional label. Hence no IP header is exposed within an MPLS domain.

Carriers may first design their own QoS PHB and codepoint scheme before they worry about interconnection. PHB and corresponding codepoint schemes usually differ between different carriers. PHBs may be mapped. A DSCP rewrite should be expected at an interconnection interface at least for plain IP traffic.

RFC3270 suggests deployment of the Short Pipe Model only in the case of VPNs. State of the art deployments also support transport of non tunneled IPv4 traffic. This is shown in figure 2.



Short-Pipe / Pen-ultimate hop popping example

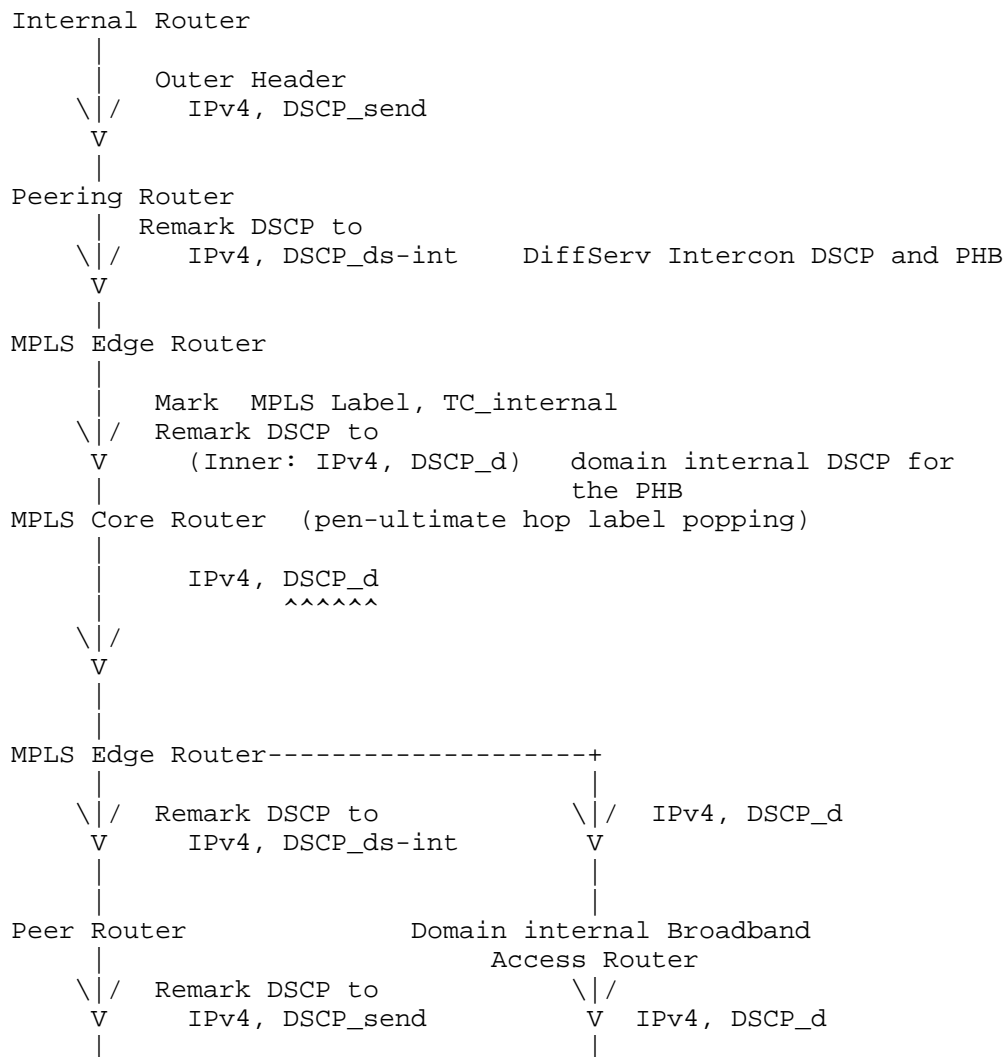
Figure 2

The packets IP DSCP must be in a well understood Diffserv context for schedulers and classifiers on the interfaces of the ultimate MPLS link. These are domain internal and a domain operating in this mode enforces DSCPs resulting in reliable domain internal QoS operation.

Without DiffServ-Intercon treatment, the traffic always leaves the domain having internal DS codepoints. DSCP_send of the figure above is remarked to the receiving domains DiffServ scheme. It leaves the

domain marked by the domains DSCP_d. Every carrier must deploy per peer PHB and DSCP mapping schemes.

If DiffServ-Intercon is applied, only traffic terminating within a domain must be aligned with the domain internal DiffServ Codepoint scheme. Traffic transiting through the domain can be easily mapped and remapped to an original DSCP. This is shown in figure 3. Of course the domain internal limitations caused by the Short Pipe model still apply.



Short-Pipe example with Diffserv-Intercon

Figure 3

Picking up terminology of RFC2983 and RFC3270, DiffServ intercon emulates the long pipe model for the PHBs it supports, if traffic is terminating in the receiving domain.

Looking at the peering interfaces only, for transiting QoS traffic DiffServ-Intercon emulates the uniform model for the PHBs and DSCPs

supported. Packets are expected to leave a domain with the DSCP/PHB as received (and per flow within each PHB in the same order as received). MPLS Treatment Aggregates should not experience congestion under standard operational conditions. The peering links need to be engineered to be congestion free too for QoS PHBs, if also the IP transit transport is to be congestion free.

Appendix C. Change log

- 00 to 01 Added terminology and references. Added details and information to interconnection class and codepoint scheme. Editorial changes.
- 01 to 02 Added some references regarding related work. Clarified class definitions. Further editorial improvements.
- 02 to 03 Consistent terminology. Discussion of Network Management PHB at interconnection interfaces. Editorial review.
- 03 to 04 Again improved terminology. Better wording of Network Control PHB at interconnection interfaces.
- 04 to 05 Large rewrite and re-ordering of contents.
- 05 to 06 Description of IP and MPLS related requirements and constraints on DSCP rewrites.
- 06 to 07 Largely rewrite, improved match and comparison with RFCs 4594 and 5127.
- 07 to 08 Added Annex A and B which were forgotten when putting together -07

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
EMC Corporation
176 South Street
Hopkinton, MA
USA

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

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

G. Fairhurst
University of Aberdeen
M. Welzl
University of Oslo
November 24, 2015

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

Abstract

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

Status of This Memo

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 27, 2016.

Copyright Notice

Copyright (c) 2015 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. Terminology	4
2. Benefit of using ECN to avoid Congestion Loss	5
2.1. Improved Throughput	5
2.2. Reduced Head-of-Line Blocking	6
2.3. Reduced Probability of RTO Expiry	6
2.4. Applications that do not Retransmit Lost Packets	7
2.5. Making Incipient Congestion Visible	8
2.6. Opportunities for new Transport Mechanisms	8
3. Network Support for ECN	9
3.1. The ECN Field	10
3.2. Forwarding ECN-Capable IP Packets	10
3.3. Enabling ECN in Network Devices	10
3.4. Co-existence of ECN and non-ECN flows	11
3.5. Bleaching and Middlebox Requirements to deploy ECN	11
3.6. Tunneling ECN and the use of ECN by Lower Layer Networks	12
4. Using ECN across the Internet	12
4.1. Partial Deployment	13
4.2. Detecting whether a Path Really Supports ECN	13
4.3. Detecting ECN Receiver Feedback Cheating	14
5. Summary: Enabling ECN in Network Devices and Hosts	14
6. Acknowledgements	15
7. IANA Considerations	16
8. Security Considerations	16
9. Revision Information	16
10. References	18
10.1. Normative References	18
10.2. Informative References	18
Authors' Addresses	20

1. Introduction

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

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

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

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

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

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

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

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

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

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

1.1. Terminology

The following terms are used:

AQM: Active Queue Management.

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

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

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

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

to provide end-to-end congestion control.

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

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

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

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

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

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

2. Benefit of using ECN to avoid Congestion Loss

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

2.1. Improved Throughput

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

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

2.2. Reduced Head-of-Line Blocking

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

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

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

2.3. Reduced Probability of RTO Expiry

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

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

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

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

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

2.4. Applications that do not Retransmit Lost Packets

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

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

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

2.5. Making Incipient Congestion Visible

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

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

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

2.6. Opportunities for new Transport Mechanisms

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

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

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

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

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

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

3. Network Support for ECN

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

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

- o ECN can be incrementally deployed in the general Internet. The IETF has provided guidance on configuration and usage in [RFC7567].
- o ECN may be deployed within a controlled environment, for example within a data centre or within a well-managed private network. This use of ECN may be tuned to the specific use-case. An example is DCTCP [AL10] [ID.DCTCP].

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

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

3.1. The ECN Field

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

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

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

3.2. Forwarding ECN-Capable IP Packets

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

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

3.3. Enabling ECN in Network Devices

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

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

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

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

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

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

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

3.5. Bleaching and Middlebox Requirements to deploy ECN

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

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

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

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

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

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

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

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

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

4. Using ECN across the Internet

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

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

4.1. Partial Deployment

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

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

4.2. Detecting whether a Path Really Supports ECN

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

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

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

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

4.3. Detecting ECN Receiver Feedback Cheating

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

5. Summary: Enabling ECN in Network Devices and Hosts

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

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

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

Table 2: Summary of Key Benefits

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

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

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

- o A network device may support alternate ECN semantics (see Section 3.1).
- o A network device must not choose a different network path solely because a packet carries has a CE-codepoint set in the ECN Field, CE-marked packets need to follow the same path as packets with an ECT(0) or ECT(1) codepoint (see Section 3.1). Network devices need to be configured not to drop packets solely because the ECT(0) or ECT(1) codepoints are used (see Section 3.2).
- o A network device must not change a packet with a CE mark to a not-ECN-capable codepoint ('00'), if the network device decides not to forward the packet with the CE-mark, it has to instead drop the packet and not bleach the marking (see Section 3.5).
- o An ECN-capable network device should correctly update the ECN codepoint of ECN-capable packets in the presence of incipient congestion (see Section 3.3).
- o Network devices need to be able to forward both ECN-capable and not-ECN-capable flows (see Section 3.4).

Prerequisites for network endpoints to enable use of ECN include:

- o An application should use an Internet transport that can set and receive ECN marks (see Section 4).
- o An ECN-capable transport/application must return feedback indicating congestion to the sending endpoint and perform an appropriate congestion response (see Section 4).
- o An ECN-capable transport/application should detect paths where there is there is persistent misuse of ECN and fall back to not sending ECT(0) or ECT(1) (see Section 4.2).
- o Designers of applications/transport are encouraged to include mechanisms that can detect and react appropriately to misbehaving receivers that fail to report CE-marked packets (see Section 4.3).

6. Acknowledgements

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

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

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

7. IANA Considerations

XX RFC Ed - PLEASE REMOVE THIS SECTION XXX

This memo includes no request to IANA.

8. Security Considerations

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

9. Revision Information

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

Revision 00 was the first WG draft.

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

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

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

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

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

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

Revision 04 (WG Review during WGLC)

Updated the abstract.

Added a table of contents.

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

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

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

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

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

Revision 05 (after WGLC comments)

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

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

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

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

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

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

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

Revision 06

Corrections from Miria.

Revision 07

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

Revision 08

Typo and definition correction from Bob Briscoe.

10. References

10.1. Normative References

- [ID.RFC5405.bis] Eggert, Lars., Fairhurst, Gorry., and Greg. Shepherd, "Unicast UDP Usage Guidelines", 2015.
- [RFC2474] "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers".
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<http://www.rfc-editor.org/info/rfc3168>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<http://www.rfc-editor.org/info/rfc6040>>.
- [RFC7567] Baker, F. and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management", Internet-draft draft-ietf-aqm-recommendation-06, October 2014.

10.2. Informative References

- [AL10] Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and M. Sridharan, "Data Center TCP (DCTCP)", SIGCOMM 2010, August 2010.
- [BA11] Bauer, Steven., Beverly, Robert., and Arthur. Berger, "Measuring the State of ECN Readiness in Servers, Clients, and Routers, ACM IMC", 2011.

- [Fla13] Flach, Tobias., Dukkipati, Nandita., Terzis, Andreas., Raghavan, Barath., Cardwell, Neal., Cheng, Yuchung., Jain, Ankur., Hao, Shuai., Katz-Bassett, Ethan., and Ramesh. Govindan, "Reducing web latency: the virtue of gentle aggression.", SIGCOMM 2013, October 2013.
- [ID.Acc.ECN] Briscoe, Bob., Scheffeneger, Richard., and Mirja. Kuehlewind, "More Accurate ECN Feedback in TCP, Work-in-Progress".
- [ID.AQM.eval] Kuhn, Nicolas., Natarajan, Preethi., Ros, David., and Naeem. Khademi, "AQM Characterization Guidelines (Work-in-progress, draft-ietf-aqm-eval-guidelines)", 2015.
- [ID.DCTCP] Bensley, S., Eggert, Lars., and D. Thaler, "Microsoft's Datacenter TCP (DCTCP): TCP Congestion Control for Datacenters (Work-in-progress, draft-bensley-tcpm-dctcp)", 2015.
- [ID.ECN-Encap] Briscoe, B., Kaippallimalil, J., and P. Thaler, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP", Internet-draft, IETF work-in-progress draft-ietf-tsvwg-ecn-encap-guidelines.
- [ID.Fallback] Kuehlewind, Mirja. and Brian. Trammell, "A Mechanism for ECN Path Probing and Fallback, draft-kuehlewind-tcpm-ecn-fallback, Work-in-Progress".
- [RFC0768] Postel, J., "User Datagram Protocol", 1980.
- [RFC1349] "Type of Service in the Internet Protocol Suite".
- [RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, DOI 10.17487/RFC3649, December 2003, <<http://www.rfc-editor.org/info/rfc3649>>.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <<http://www.rfc-editor.org/info/rfc3758>>.

- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <<http://www.rfc-editor.org/info/rfc4340>>.
- [RFC4774] Floyd, S., "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field", BCP 124, RFC 4774, DOI 10.17487/RFC4774, November 2006, <<http://www.rfc-editor.org/info/rfc4774>>.
- [RFC5562] Kuzmanovic, A., Mondal, A., Floyd, S., and K. Ramakrishnan, "Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets", RFC 5562, DOI 10.17487/RFC5562, June 2009, <<http://www.rfc-editor.org/info/rfc5562>>.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<http://www.rfc-editor.org/info/rfc5681>>.
- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <<http://www.rfc-editor.org/info/rfc6679>>.
- [RFC6789] Briscoe, B., Ed., Woundy, R., Ed., and A. Cooper, Ed., "Congestion Exposure (ConEx) Concepts and Use Cases", RFC 6789, DOI 10.17487/RFC6789, December 2012, <<http://www.rfc-editor.org/info/rfc6789>>.
- [ST14] Stewart, R., Tuexen, M., and X. Dong, "ECN for Stream Control Transmission Protocol (SCTP)", Internet-draft draft-stewart-tsvwg-sctpecn-05.txt, January 2014.
- [TR15] Trammell, Brian., Kuehlewind, Mirja., Boppart, Damiano, Learmonth, Iain., and Gorry. Fairhurst, "Enabling internet-wide deployment of Explicit Congestion Notification Tramwell, B., Kuehlewind, M., Boppart, D., Learmonth, I., Fairhurst, G. & Scheffnegger, Passive and Active Measurement Conference (PAM)", March 2015.

Authors' Addresses

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

Email: gorry@erg.abdn.ac.uk

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

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

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: March 13, 2014

R. Stewart
Adara Networks
M. Tuexen
I. Ruengeler
Muenster Univ. of Appl. Sciences
September 09, 2013

Stream Control Transmission Protocol (SCTP) Network Address Translation
draft-ietf-behave-sctpnat-09.txt

Abstract

Stream Control Transmission Protocol [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 for TCP that allows multiple hosts to reside behind a NAT and yet use only a single globally unique IPv4 address, even when two hosts (behind a NAT) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation or NAPT. To date, specialized code for SCTP has NOT yet been added to most NATs so that only pure NAT is available. The end result of this is that only one SCTP capable host can be behind a NAT.

This document describes an SCTP specific variant of NAT which provides similar features of NAPT in the single point and multi-point traversal scenario.

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 13, 2014.

Copyright Notice

Copyright (c) 2013 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	3
3. Terminology	3
4. SCTP NAT Traversal Scenarios	4
4.1. Single Point Traversal	4
4.2. Multi Point Traversal	5
5. Limitations of Classical NAPT for SCTP	6
6. The SCTP Specific Variant of NAT	6
7. NAT to SCTP	10
8. Handling of Fragmented SCTP Packets	10
9. Various Examples of NAT Traversals	10
9.1. Single-homed Client to Single-homed Server	10
9.2. Single-homed Client to Multi-homed Server	12
9.3. Multihomed Client and Server	15
9.4. NAT Loses Its State	18
9.5. Peer-to-Peer Communication	20
10. IANA Considerations	24
11. Security Considerations	24
12. Acknowledgments	24
13. References	24
13.1. Normative References	24
13.2. Informative References	25
Authors' Addresses	25

1. Introduction

Stream Control Transmission Protocol [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 for TCP that allows multiple hosts to reside behind a NAT and use private

addresses (see [RFC5735]) and yet use only a single globally unique IPv4 address, even when two hosts (behind a NAT) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation or NAPT. To date, specialized code for SCTP has not yet been added to most NATs so that only true NAT is available. The end result of this is that only one SCTP capable host can be behind a NAT.

This document proposes an SCTP specific variant NAT that provides the NAPT functionality without changing SCTP port numbers. The authors feel it is possible and desirable to make these changes for a number of reasons.

- o It is desirable for SCTP internal end-hosts on multiple platforms to be able to share a NAT's public IP address, much as TCP does today.
- o If a NAT 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 required by SCTP.
- o Not having to touch the IP payload makes the processing of ICMP messages in NATs easier.

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. Terminology

For this discussion we will use several terms, which we will define and point out in Figure 1.

Private-Address (Priv-Addr): The private address that is known to the internal host.

Internal-Port (Int-Port): The port number that is in use by the host holding the Private-Address.

Internal-VTag (Int-VTag): The Verification Tag that the internal host has chosen for its communication. The VTag is a unique 32 bit tag that must accompany any incoming SCTP packet for this association to the Private-Address.

External-Address (Ext-Addr): The address that an internal host is attempting to contact.

External-Port (Ext-Port): The port number of the peer process at the External-Address.

External-VTag (Ext-VTag): The Verification Tag that the host holding the External-Address has chosen for its communication. The VTag is a unique 32 bit tag that must accompany any incoming SCTP packet for this association to the External-Address.

Public-Address (Pub-Addr): The public address assigned to the NAT box which it uses as a source address when sending packets towards the External-Address.

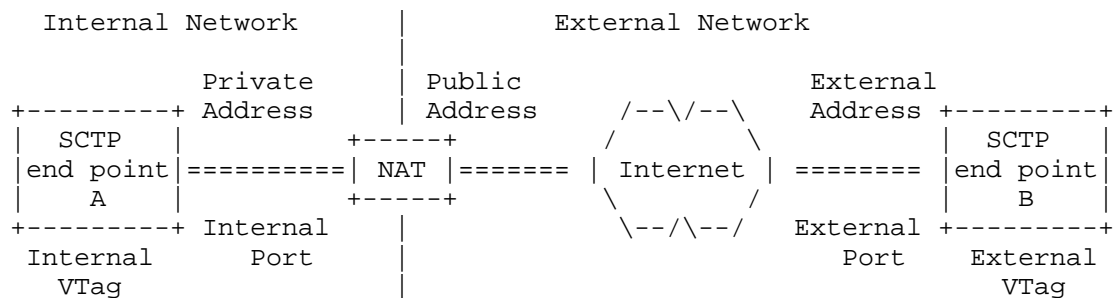


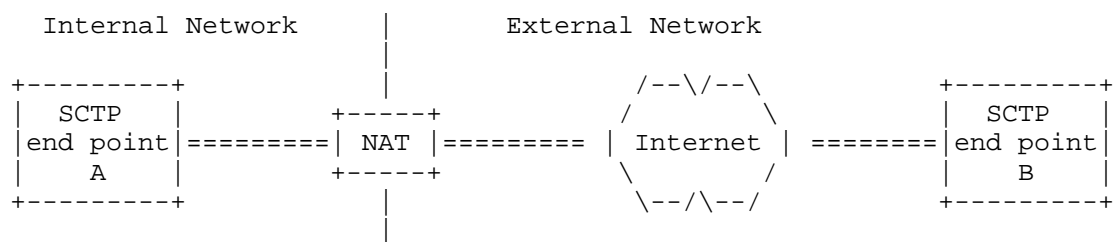
Figure 1: Architecture

4. SCTP NAT Traversal Scenarios

This section defines the notion of single and multi-point NAT traversal.

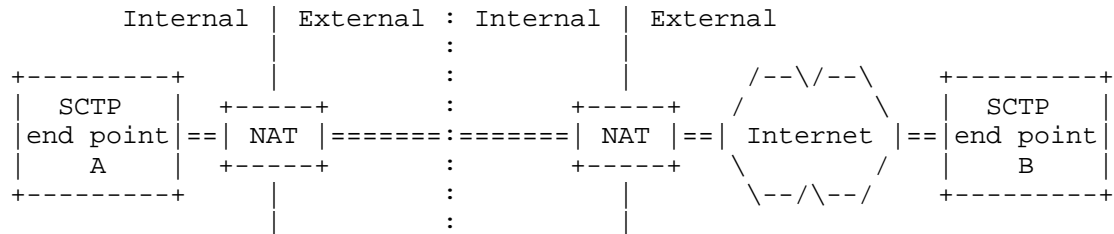
4.1. Single Point Traversal

In this case, all packets in the SCTP association go through a single NAT, as shown below:



Single NAT scenario

A variation of this case is shown below, i.e., multiple NATs in a single path:



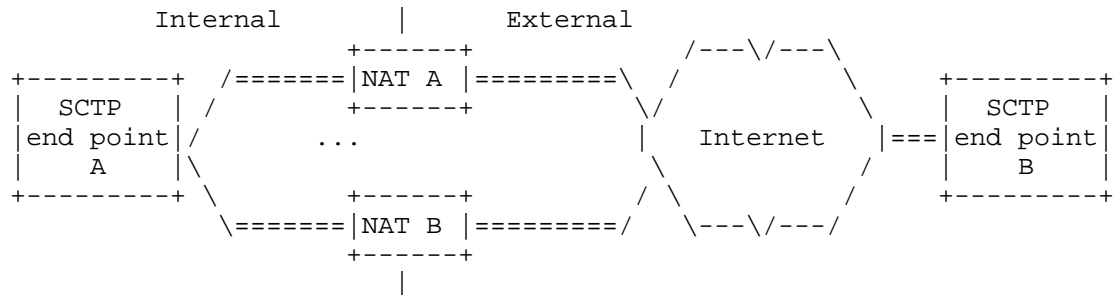
Serial NATs scenario

In this single point traversal scenario, we must acknowledge that while one of the main benefits of SCTP multi-homing is redundant paths, the NAT function represents a single point of failure in the path of the SCTP multi-home association. However, the rest of the path may 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 (or NATs) in this case sees all the packets of the SCTP association.

4.2. Multi Point Traversal

This case involves multiple NATs and each NAT only sees some of the packets in the SCTP association. An example is shown below:



Parallel NATs 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.

5. Limitations of Classical NAT for SCTP

Using classical NAT may result in changing one of the SCTP port numbers during the processing which requires the recomputation of the transport layer checksum. Whereas for UDP and TCP this can be done very efficiently, for SCTP the checksum (CRC32c) over the entire packet needs to be recomputed. This would add considerable to the NAT computational burden, however hardware support may mitigate this in some implementations.

An SCTP endpoint may have multiple addresses but only has a single port number. To make multipoint traversal work, all the NATs involved must 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 pre-defined table of ports and addresses configured within each NAT. Other mechanisms could make use of NAT to NAT communication. Such mechanisms are considered by the authors not to be deployable on a wide scale base and thus not a recommended solution. Therefore the SCTP variant of NAT has been developed.

6. The SCTP Specific Variant of NAT

In this section we assume that we have multiple SCTP capable hosts behind a NAT which has one Public-Address. Furthermore we are focusing in this section on the single point traversal scenario.

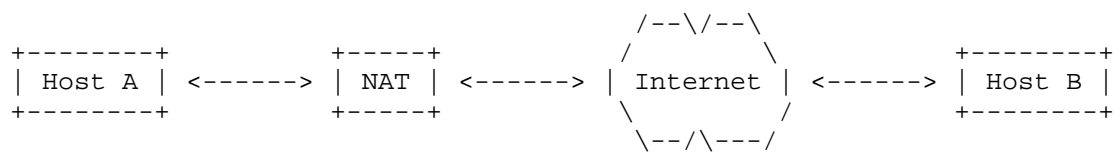
The modification of SCTP packets sent to the public Internet is easy. The source address of the packet has to be replaced with the Public-Address. It may also be necessary to establish some state in the NAT box to handle incoming packets, which is discussed later.

For SCTP packets coming from the public Internet the destination address of the packets has to be replaced with the Private-Address of the host the packet has to be delivered to. The lookup of the Private-Address is based on the External-VTag, External-Port, External-Address, Internal-VTag and the Internal-Port.

For the SCTP NAT processing the NAT box has to maintain a table of Internal-VTag, Internal-Port, Private-Address, External-VTag, External-Port and whether the restart procedure is disabled or not. An entry in that table is called a NAT state control block. The function Create() obtains the just mentioned parameters and returns a NAT-State control block.

The entries in this table fulfill some uniqueness conditions. There must not be more than one entry with the same pair of Internal-Port and External-Port. This rule can be relaxed, if all entries with the same Internal-Port and External-Port have the support for the restart procedure enabled. In this case there must be no more than one entry with the same Internal-Port, External-Port and Ext-VTag and no more than one entry with the same Internal-Port, External-Port and Int-VTag.

The processing of outgoing SCTP packets containing an INIT-chunk is described in the following figure. The scenario shown is valid for all message flows in this section.



```

                INIT[Initiate-Tag]
Priv-Addr:Int-Port -----> Ext-Addr:Ext-Port
                        Ext-VTag=0

                Create(Initiate-Tag, Int-Port, Priv-Addr, 0)
                Returns(NAT-State control block)

Translate To:

                INIT[Initiate-Tag]
Pub-Addr:Int-Port -----> Ext-Addr:Ext-Port
                        Ext-VTag=0

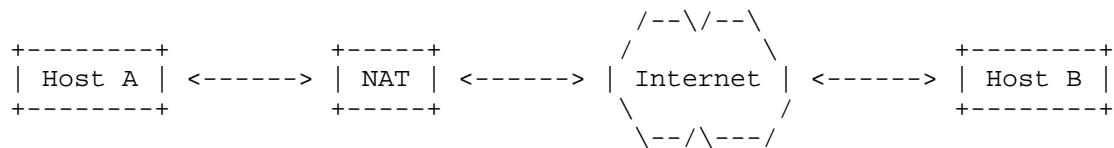
```

It should be noted that normally a NAT control block will be created. However, it is possible that there is already a NAT control block with the same External-Address, External-Port, Internal-Port, and Internal-VTag but different Private-Address. In this case the INIT SHOULD be dropped by the NAT and an ABORT SHOULD be sent back to the SCTP host with the M-Bit set and an appropriate error cause (see [I-D.ietf-tsvwg-natsupp] 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.

It is also possible that a connection to External-Address and External-Port exists without an Internal-VTag conflict but the

External-Address does not support the DISABLE_RESTART feature (noted in the NAT control block when the prior connection was established). In such a case the INIT SHOULD be dropped by the NAT and an ABORT SHOULD be sent back to the SCTP host with the M-Bit set and an appropriate error cause (see [I-D.ietf-tsvwg-natsupp] for the format).

The processing of outgoing SCTP packets containing no INIT-chunk is described in the following figure.

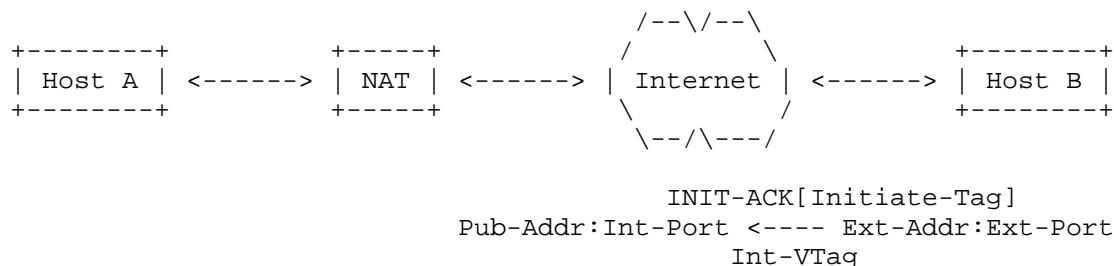


Priv-Addr:Int-Port -----> Ext-Addr:Ext-Port
 Ext-VTag

Translate To:

Pub-Addr:Int-Port -----> Ext-Addr:Ext-Port
 Ext-VTag

The processing of incoming SCTP packets containing INIT-ACK chunks is described in the following figure. The Lookup() function getting as input the Internal-VTag, Internal-Port, External-VTag (=0), External-Port, and External-Address, returns the corresponding entry of the NAT table and updates the External-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.




```

Lookup(Int-VTag, Int-Port, *, 0, Ext-Port)
Update(*, *, *, Initiate-Tag, *)

```

```

Returns(NAT-State control block containing Private-Address)

```

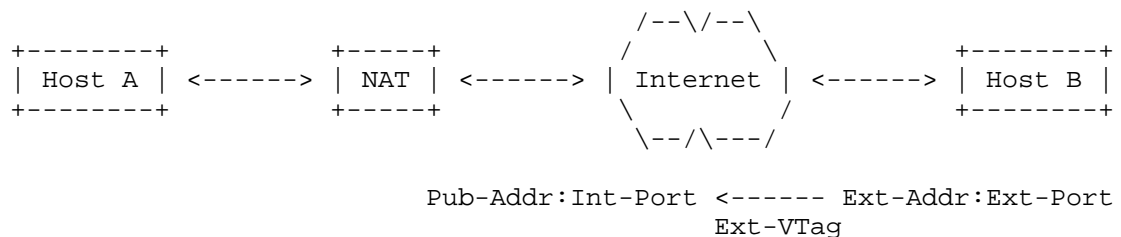
```

      INIT-ACK[Initiate-Tag]
Priv-Addr:Int-Port <----- Ext-Addr:Ext-Port
                    Int-VTag

```

In the case Lookup fails, the SCTP packet is dropped. The Update routine inserts the External-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 described in the following figure.



```

Lookup(0, Int-Port, *, Ext-VTag, Ext-Port)

```

```

Returns(NAT-State control block containing Private-Address)

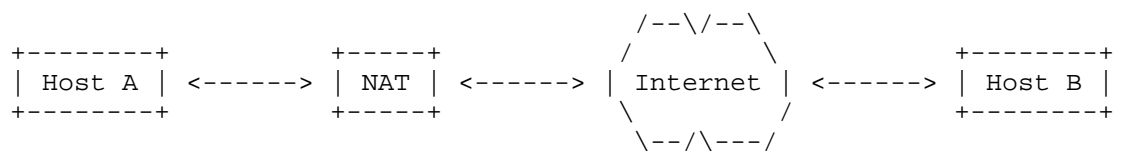
```

```

Priv-Addr:Int-Port <----- Ext-Addr:Ext-Port
                    Ext-VTag

```

The processing of other incoming SCTP packets is described in the following figure.



```

Pub-Addr: Int-Port <----- Ext-Addr: Ext-Port
                          Int-VTag

```

```

Lookup(Int-VTag, Int-Port, *, *, Ext-Port)

```

```

Returns(NAT-State control block containing Local-Address)

```

```

Priv-Addr: Int-Port <----- Ext-Addr: Ext-Port
                          Int-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 an External-VTag of zero is found, it is considered a match and the External-VTag is updated.

This allows the handling of INIT-collision through NAT.

7. NAT to SCTP

This document at various places discusses the sending of specialized SCTP chunks (e.g. an ABORT with M-Bit set). These chunks and procedures are not defined in this document, but instead are defined in [I-D.ietf-tsvwg-natsupp]. The NAT implementer should refer to [I-D.ietf-tsvwg-natsupp] for detailed descriptions of packet formats and procedures.

8. Handling of Fragmented SCTP Packets

A NAT box MUST support IP reassembly of received fragmented SCTP packets. The fragments may arrive in any order.

When an SCTP packet has to be fragmented by the NAT box and the IP header forbids fragmentation a corresponding ICMP packet SHOULD be sent.

9. Various Examples of NAT Traversals

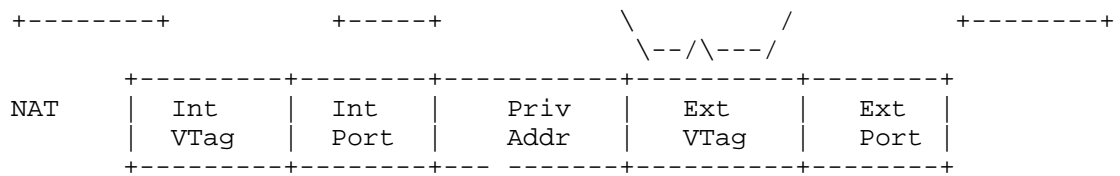
9.1. Single-homed Client to Single-homed Server

The internal client starts the association with the external server via a four-way-handshake. Host A starts by sending an INIT chunk.

```

+-----+           +-----+           /--\ /--\           +-----+
| Host A | <-----> | NAT   | <-----> | Internet | <-----> | Host B |

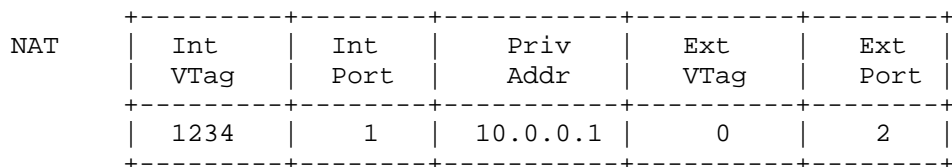
```



```
INIT[Initiate-Tag = 1234]
10.0.0.1:1 -----> 100.0.0.1:2
    Ext-VTtag = 0
```

A NAT entry is created, the source address is substituted and the packet is sent on:

NAT creates entry:

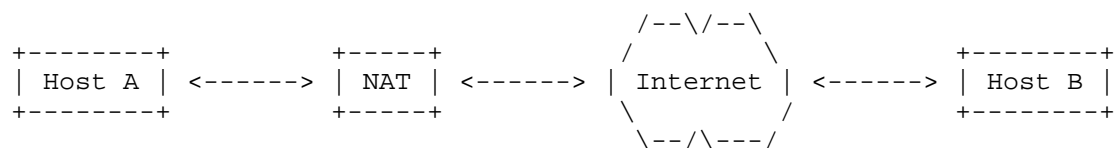


```

INIT[Initiate-Tag = 1234]
101.0.0.1:1 -----> 100.0.0.1:2
                     Ext-VTtag = 0

```

Host B receives the INIT and sends an INIT-ACK with the NAT's external address as destination address.

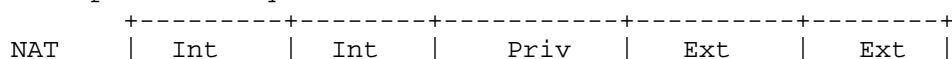


```

          INIT-ACK[Initiate-Tag = 5678]
101.0.0.1:1 <----- 100.0.0.1:2
                  Int-VTag = 1234

```

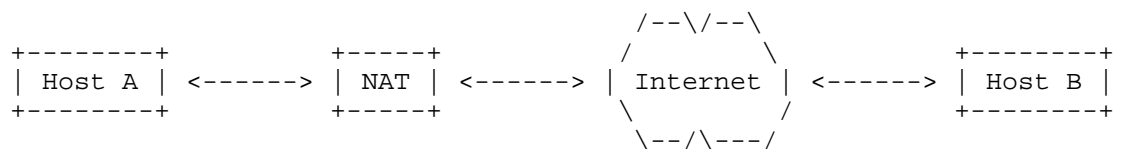
NAT updates entry:



VTag	Port	Addr	VTag	Port
1234	1	10.0.0.1	5678	2

```
INIT-ACK[Initiate-Tag = 5678]
10.0.0.1:1 <----- 100.0.0.1:2
      Int-VTag = 1234
```

The handshake finishes with a COOKIE-ECHO acknowledged by a COOKIE-ACK.



```
      COOKIE-ECHO
10.0.0.1:1 -----> 100.0.0.1:2
      Ext-VTag = 5678
```

```

                                COOKIE-ECHO
101.0.0.1:1 -----> 100.0.0.1:2
                                Ext-VTag = 5678
```

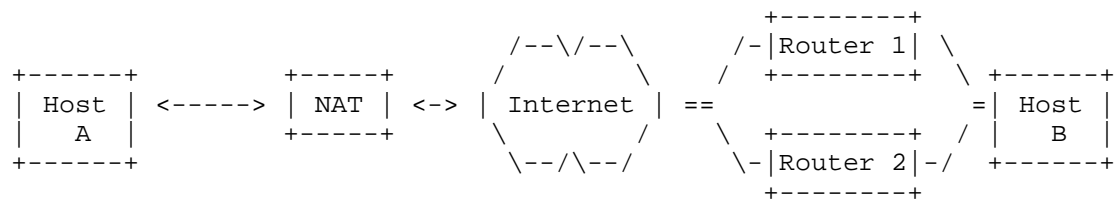
```

                                COOKIE-ACK
101.0.0.1:1 <----- 100.0.0.1:2
                                Int-VTag = 1234
```

```
      COOKIE-ACK
10.0.0.1:1 <----- 100.0.0.1:2
      Int-VTag = 1234
```

9.2. Single-homed Client to Multi-homed Server

The internal client is single-homed whereas the external server is multi-homed. The client (Host A) sends an INIT like in the single-homed case.



NAT	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port

```

INIT[Initiate-Tag = 1234]
10.0.0.1:1 ---> 100.0.0.1:2
      Ext-VTag = 0

```

NAT creates entry:

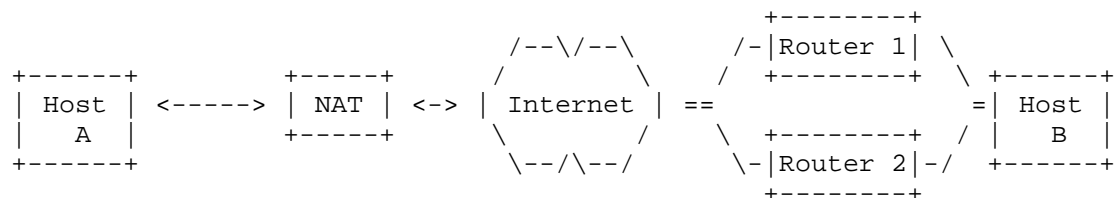
NAT	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.0.0.1	0	2

```

                        INIT[Initiate-Tag = 1234]
101.0.0.1:1 -----> 100.0.0.1:2
                        Ext-VTag = 0

```

The server (Host B) includes its two addresses in the INIT-ACK chunk, which results in two NAT entries.



```

      INIT-ACK[Initiate-tag = 5678, IP-Addr = 100.1.0.1]
101.0.0.1:1 <----- 100.0.0.1:2
                  Int-VTag = 1234

```

NAT does need to change the table for second address:

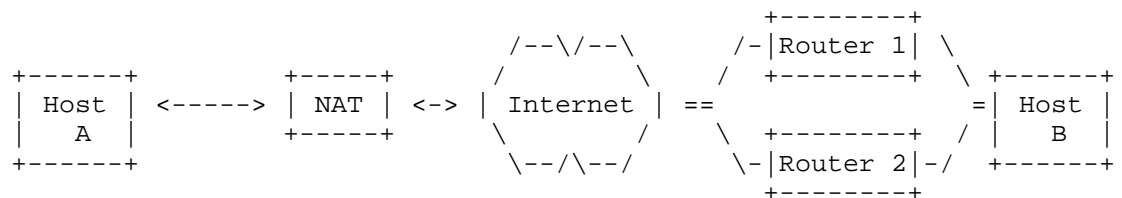
NAT						
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	
	1234	1	10.0.0.1	5678	2	

```

INIT-ACK[Initiate-Tag = 5678]
10.0.0.1:1 <--- 100.0.0.1:2
      Int-VTag = 1234

```

The handshake finishes with a COOKIE-ECHO acknowledged by a COOKIE-ACK.



```

      COOKIE-ECHO
10.0.0.1:1 ---> 100.0.0.1:2
      ExtVTag = 5678

```

```

                                COOKIE-ECHO
101.0.0.1:1 -----> 100.0.0.1:2
                                Ext-VTag = 5678

```

```

                                COOKIE-ACK
101.0.0.1:1 <----- 100.0.0.1:2
                                Int-VTag = 1234

```

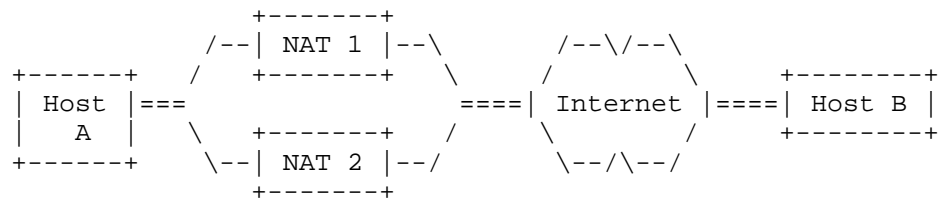
```

      COOKIE-ACK
10.0.0.1:1 <--- 100.0.0.1:2
      Int-VTag = 1234

```

9.3. Multihomed Client and Server

The client (Host A) sends an INIT to the server (Host B), but does not include the second address.



NAT 1	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port

```

      INIT[Initiate-Tag = 1234]
10.0.0.1:1 -----> 100.0.0.1:2
      Ext-VTag = 0

```

NAT 1 creates entry:

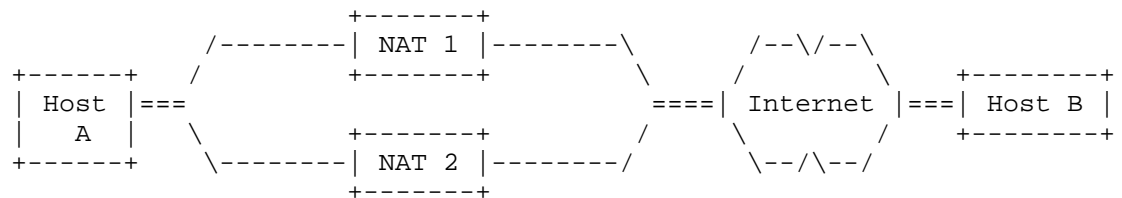
NAT 1	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.0.0.1	0	2

```

      INIT[Initiate-Tag = 1234]
101.0.0.1:1 -----> 100.0.0.1:2
      ExtVTag = 0

```

Host B includes its second address in the INIT-ACK, which results in two NAT entries in NAT 1.



```

INIT-ACK[Initiate-Tag = 5678, IP-Addr = 100.1.0.1]
101.0.0.1:1 <----- 100.0.0.1:2
                    Int-VTag = 1234

```

NAT 1 does not need to update the table for second address:

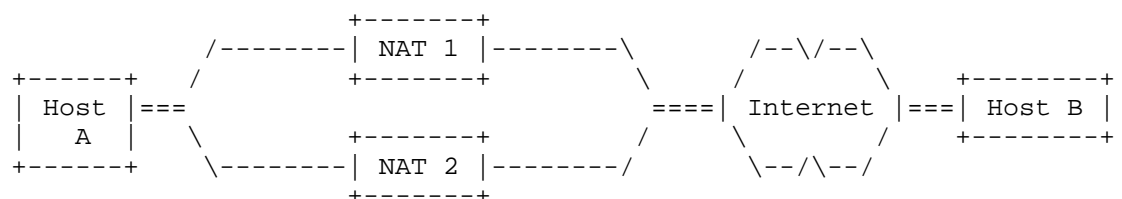
NAT 1					
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.0.0.1	5678	2

```

INIT-ACK[Initiate-Tag = 5678]
10.0.0.1:1 <-----100.0.0.1:2
                    Int-VTag = 1234

```

The handshake finishes with a COOKIE-ECHO acknowledged by a COOKIE-ACK.



COOKIE-ECHO


```
10.0.0.1:1 -----> 100.0.0.1:2
      Ext-VTag = 5678
```

```

                                COOKIE-ECHO
101.0.0.1:1 -----> 100.0.0.1:2
      Ext-VTag = 5678
```

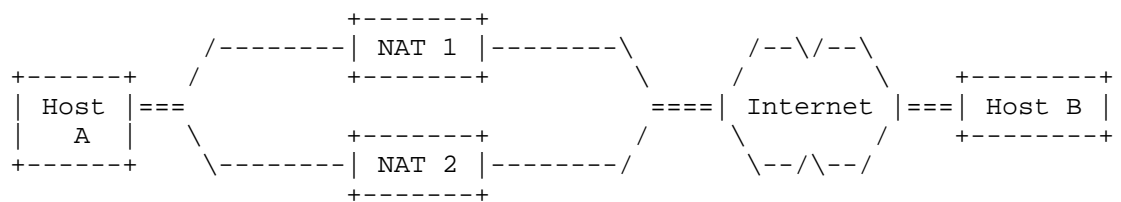
```

                                COOKIE-ACK
101.0.0.1:1 <----- 100.0.0.1:2
      Int-VTag = 1234
```

```

                                COOKIE-ACK
10.0.0.1:1 <----- 100.0.0.1:2
      Int-VTag = 1234
```

Host A announces its second address in an ASCONF chunk. The address parameter contains an undefined address (0) to indicate that the source address should be added. The lookup address parameter within the ASCONF chunk will also contain the pair of VTags (external and internal) so that the NAT may populate its table completely with this single packet.



```

ASCONF [ADD-IP=0.0.0.0, INT-VTag=1234, Ext-VTag = 5678]
10.1.0.1:1 -----> 100.1.0.1:2
      Ext-VTag = 5678
```

NAT 2 creates complete entry:

NAT 2					
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.1.0.1	5678	2

```

+-----+-----+-----+-----+-----+
ASCONF [ADD-IP,Int-VTag=1234, Ext-VTag = 5678]
101.1.0.1:1 -----> 100.1.0.1:2
                        Ext-VTag = 5678

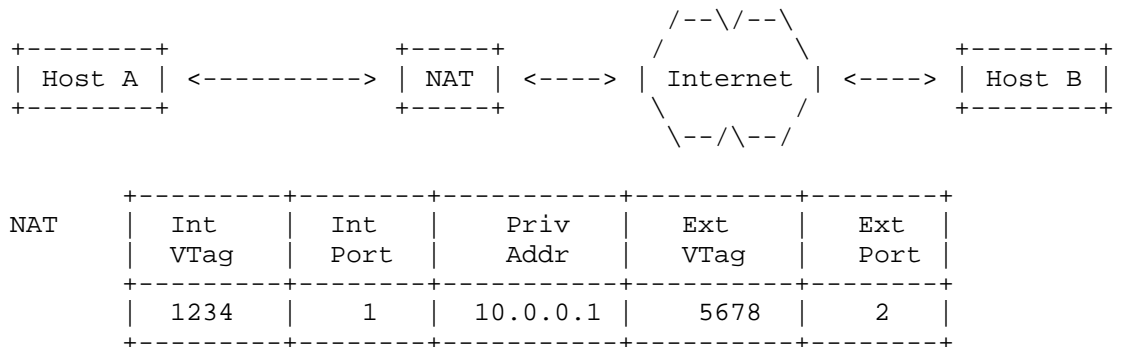
                        ASCONF-ACK
101.1.0.1:1 <----- 100.1.0.1:2
                        Int-VTag = 1234

ASCONF-ACK
10.1.0.1:1 <----- 100.1.0.1:2
                        Int-VTag = 1234

```

9.4. NAT Loses Its State

Association is already established between Host A and Host B, when the NAT loses its state and obtains a new public address. Host A sends a DATA chunk to Host B.



```

DATA
10.0.0.1:1 -----> 100.0.0.1:2
                        Ext-VTag = 5678

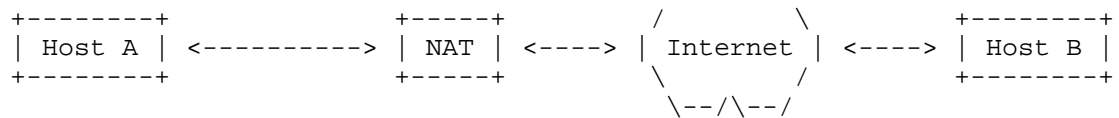
```

The NAT box cannot find entry for the association. It sends ERROR message with the M-Bit set and the cause "NAT state missing".

```

/--\ /--\

```

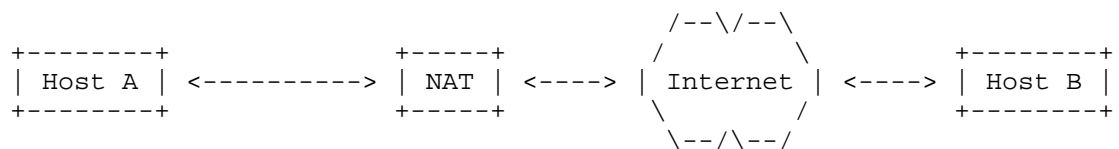


```

    ERROR [M-Bit, NAT state missing]
10.0.0.1:1 <----- 100.0.0.1:2
      Ext-VTag = 5678

```

On reception of the ERROR message, Host A sends 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, Ext-VTag = 5678]
10.0.0.1:1 -----> 100.1.0.1:2
      Ext-VTag = 5678

```

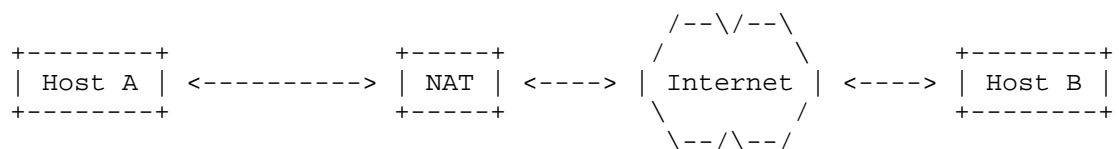
NAT						
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	
	1234	1	10.0.0.1	5678	2	

```

ASCONF [ADD-IP,DELETE-IP,Int-VTag=1234, Ext-VTag = 5678]
      102.1.0.1:1 -----> 100.1.0.1:2
                        Ext-VTag = 5678

```

Host B adds the new source address and deletes all former entries.



```

                                ASCONF-ACK
102.1.0.1:1 <----- 100.1.0.1:2
                                Int-VTag = 1234

                                ASCONF-ACK
10.1.0.1:1 <----- 100.1.0.1:2
                                Int-VTag = 1234

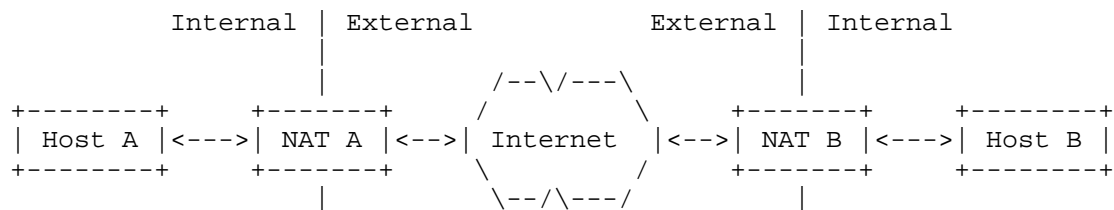
                                DATA
10.0.0.1:1 -----> 100.0.0.1:2
                                Ext-VTag = 5678

                                DATA
102.1.0.1:1 -----> 100.1.0.1:2
                                Ext-VTag = 5678

```

9.5. Peer-to-Peer Communication

If two hosts are behind NATs, they have to get knowledge of the peer's public address. This can be achieved with a so-called rendezvous server. Afterwards the destination addresses are public, and the association is set up with the help of the INIT collision. The NAT boxes create their entries according to their internal peer's point of view. Therefore, NAT A's Internal-VTag and Internal-Port are NAT B's External-VTag and External-Port, respectively. The naming of the verification tag in the packet flow is done from the sending peer's point of view.



NAT-Tables

NAT A						
	Int	Int	Priv	Ext	Ext	
	VTag	Port	Addr	VTag	Port	
NAT B						
	Int	Int	Priv	Ext	Ext	
	v-tag	port	addr	v-tag	port	

```

+-----+-----+--- +-----+-----+
INIT[Initiate-Tag = 1234]
10.0.0.1:1 --> 100.0.0.1:2
      Ext-VTag = 0

```

NAT A creates entry:

NAT A					
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.0.0.1	0	2

```

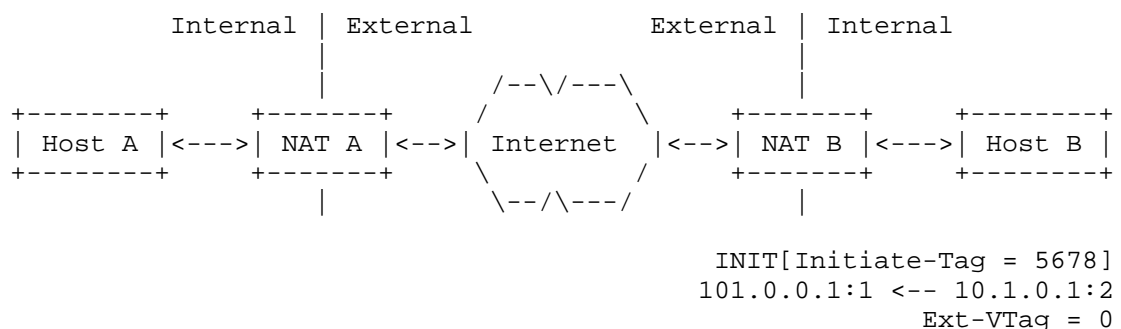
              INIT[Initiate-Tag = 1234]
101.0.0.1:1 -----> 100.0.0.1:2
              Ext-VTag = 0

```

NAT B processes INIT, but cannot find an entry. The SCTP packet is silently discarded and leaves the NAT table of NAT B unchanged.

NAT B					
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port

Now Host B sends INIT, which is processed by NAT B. Its parameters are used to create an entry.



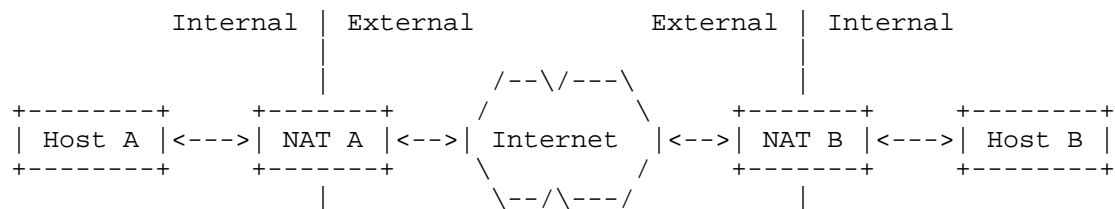
NAT B	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	5678	2	10.1.0.1	0	1

```

INIT[Initiate-Tag = 5678]
101.0.0.1:1 <----- 100.0.0.1:2
                Ext-VTag = 0

```

NAT A processes INIT. As the outgoing INIT of Host A has already created an entry, the entry is found and updated:



VTag != Int-VTag, but Ext-VTag == 0, find entry.

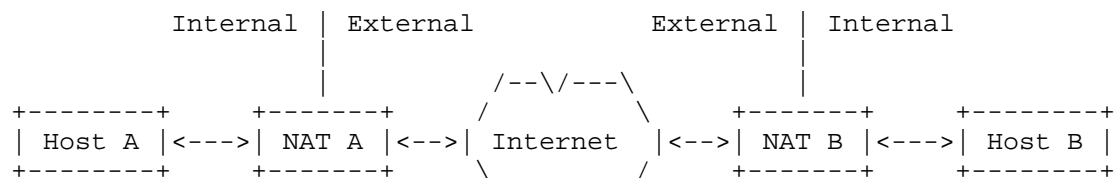
NAT A	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.0.0.1	5678	2

```

INIT[Initiate-tag = 5678]
10.0.0.1:1 <-- 100.0.0.1:2
            Ext-VTag = 0

```

Host A send INIT-ACK, which can pass through NAT B:



```

|                               \--/\---/                               |
INIT-ACK[Initiate-Tag = 1234]
10.0.0.1:1 --> 100.0.0.1:2
    Ext-VTag = 5678

```

```

INIT-ACK[Initiate-Tag = 1234]
101.0.0.1:1 -----> 100.0.0.1:2
    Ext-VTag = 5678

```

NAT B updates entry:

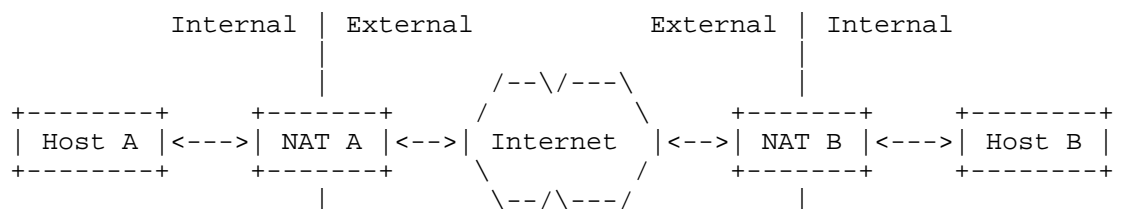
NAT B	+-----+-----+-----+-----+-----+					
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	
	5678	2	10.1.0.1	1234	1	

```

INIT-ACK[Initiate-Tag = 1234]
101.0.0.1:1 --> 10.1.0.1:2
    Ext-VTag = 5678

```

The lookup for COOKIE-ECHO and COOKIE-ACK is successful.



```

COOKIE-ECHO
101.0.0.1:1 <-- 10.1.0.1:2
    Ext-VTag = 1234

```

```

COOKIE-ECHO
101.0.0.1:1 <----- 100.0.0.1:2
    Ext-VTag = 1234

```

```

COOKIE-ECHO
10.0.0.1:1 <-- 100.0.0.1:2
    Ext-VTag = 1234

```

```
      COOKIE-ACK
10.0.0.1:1 --> 100.0.0.1:2
      Ext-VTag = 5678
```

```
      COOKIE-ACK
101.0.0.1:1 -----> 100.0.0.1:2
      Ext-VTag = 5678
```

```
      COOKIE-ACK
101.0.0.1:1 --> 10.1.0.1:2
      Ext-VTag = 5678
```

10. IANA Considerations

This document requires no actions from IANA.

11. Security Considerations

State maintenance within a NAT is always a subject of possible Denial Of Service attacks. This document recommends that at a minimum a NAT runs a timer on any SCTP state so that old association state can be cleaned up.

12. Acknowledgments

The authors wish to thank Jason But Bryan Ford, David Hayes, Alfred Hines, Henning Peters, Timo Voelker, Dan Wing, and Qiaobing Xie for their invaluable comments.

13. References

13.1. Normative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [I-D.ietf-tsvwg-natsupp]

Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control Transmission Protocol (SCTP) Network Address Translation Support", draft-ietf-tsvwg-natsupp-05 (work in progress), February 2013.

13.2. Informative References

[RFC5735] Cotton, M. and L. Vegoda, "Special Use IPv4 Addresses", RFC 5735, January 2010.

Authors' Addresses

Randall R. Stewart
Adara Networks
Chapin, SC 29036
US

Email: randall@lakerest.net

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
DE

Email: tuexen@fh-muenster.de

Irene Ruengeler
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
DE

Email: i.ruengeler@fh-muenster.de

TSVWG
Internet-Draft
Updates: 4787, 5382, 5508 (if approved)
Intended status: Best Current Practice
Expires: September 3, 2016

R. Penno
Cisco
S. Perreault
Jive Communications
M. Boucadair, Ed.
Orange
S. Sivakumar
Cisco
K. Naito
NTT
March 2, 2016

Network Address Translation (NAT) Behavioral Requirements Updates
draft-ietf-tsvwg-behave-requirements-update-08

Abstract

This document clarifies and updates several requirements of RFC4787, RFC5382, and RFC5508 based on operational and development experience. The focus of this document is NAT44.

This document updates RFCs 4787, 5382, and 5508.

Status of This Memo

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 September 3, 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.

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. Scope	3
1.2. Terminology	3
2. TCP Session Tracking	3
2.1. TCP Transitory Connection Idle-Timeout	5
2.2. TCP RST	5
3. Port Overlapping Behavior	5
4. Address Pooling Paired (APP)	6
5. Endpoint-Independent Mapping (EIM) Protocol Independence . .	7
6. Endpoint-Independent Filtering (EIF) Protocol Independence .	7
7. Endpoint-Independent Filtering (EIF) Mapping Refresh	7
7.1. Outbound Mapping Refresh and Error Packets	8
8. Port Parity	8
9. Port Randomization	8
10. IP Identification (IP ID)	9
11. ICMP Query Mappings Timeout	9
12. Hairpinning Support for ICMP Packets	9
13. IANA Considerations	9
14. Security Considerations	10
15. References	11
15.1. Normative References	11
15.2. Informative References	11
Acknowledgements	12
Contributors	13
Authors' Addresses	13

1. Introduction

[RFC4787], [RFC5382], and [RFC5508] contributed to enhance Network Address Translation (NAT) interoperability and conformance. Operational experience gained through widespread deployment and evolution of NAT indicates that some areas of the original documents need further clarification or updates. This document provides such clarifications and updates.

1.1. Scope

The goal of this document is to clarify and update the set of requirements listed in [RFC4787], [RFC5382], and [RFC5508]. The document focuses exclusively on NAT44.

The scope of this document has been set so that it does not create new requirements beyond those specified in the documents cited above.

Carrier-Grade NAT (CGN) related requirements are defined in [RFC6888].

1.2. Terminology

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

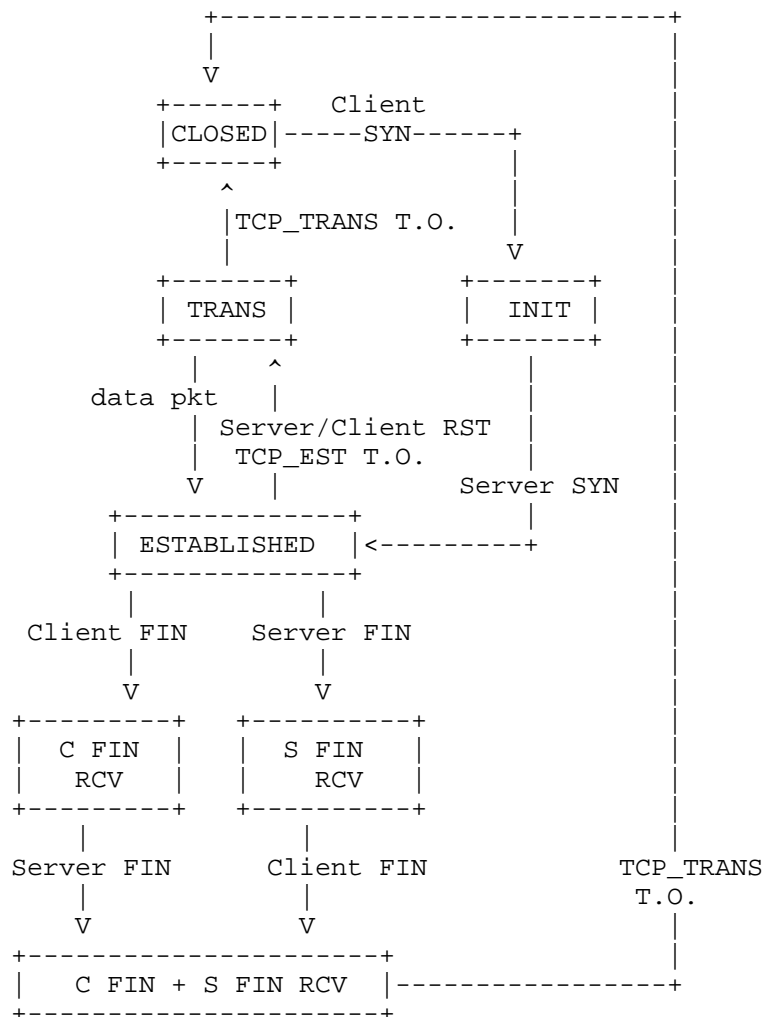
The reader is assumed to be familiar with the terminology defined in: [RFC2663],[RFC4787],[RFC5382], and [RFC5508].

In this document, the term "NAT" refers to both "Basic NAT" and "Network Address/Port Translator (NAPT)" (see Section 3 of [RFC4787]). As a reminder, Basic NAT and NAPT are two variations of traditional NAT, in that translation in Basic NAT is limited to IP addresses alone, whereas translation in NAPT is extended to include IP address and Transport identifier (such as TCP/UDP port or ICMP query ID) (refer to Section 2 of [RFC3022]).

2. TCP Session Tracking

[RFC5382] specifies TCP timers associated with various connection states but does not specify the TCP state machine a NAT44 should follow as a basis to apply such timers.

Update: The TCP state machine depicted in Figure 1, adapted from [RFC6146], SHOULD be implemented by a NAT for TCP session tracking purposes.



Legend:

- * Messages sent or received from the server are prefixed with "Server".
- * Messages sent or received from the client are prefixed with "Client".
- * "C" means "Client-side"
- * "S" means "Server-side".
- * TCP_EST T.O: refers to the established connection idle timeout as defined in [RFC5382].
- * TCP_TRANS T.O: refers to the transitory connection idle timeout as defined in [RFC5382].

Figure 1: Simplified version of the TCP State Machine

2.1. TCP Transitory Connection Idle-Timeout

The transitory connection idle-timeout is defined as the minimum time a TCP connection in the partially open or closing phases must remain idle before the NAT considers the associated session a candidate for removal (REQ-5 of [RFC5382]). But [RFC5382] does not clearly state whether these can be configured separately.

Clarification: This document clarifies that a NAT SHOULD provide different configurable parameters for configuring the open and closing idle timeouts.

To accommodate deployments that consider a partially open timeout of 4 minutes as being excessive from a security standpoint, a NAT MAY allow the configured timeout to be less than 4 minutes. However, a minimum default transitory connection idle-timeout of 4 minutes is RECOMMENDED.

2.2. TCP RST

[RFC5382] leaves the handling of TCP RST packets unspecified.

Update: This document adopts a similar default behavior as in [RFC6146]. Concretely, when the NAT receives a TCP RST matching an existing mapping, it MUST translate the packet according to the NAT mapping entry. Moreover, the NAT SHOULD wait for 4 minutes before deleting the session and removing any state associated with it if no packets are received during that 4 minutes timeout.

Notes:

- * Admittedly, the NAT has to verify whether received TCP RST packets belong to a connection. This verification check is required to avoid off-path attacks.
- * If the NAT removes immediately the NAT mapping upon receipt of a TCP RST message, stale connections may be maintained by endpoints if the first RST message is lost between the NAT and the recipient.

3. Port Overlapping Behavior

REQ-1 from [RFC4787] and REQ-1 from [RFC5382] specify a specific port overlapping behavior; that is the external IP address and port can be reused for connections originating from the same internal source IP address and port irrespective of the destination. This is known as endpoint-independent mapping (EIM).

Update: This document clarifies that this port overlapping behavior may be extended to connections originating from different internal source IP addresses and ports as long as their destinations are different.

The following mechanism MAY be implemented by a NAT:

If destination addresses and ports are different for outgoing connections started by local clients, a NAT MAY assign the same external port as the source ports for the connections. The port overlapping mechanism manages mappings between external packets and internal packets by looking at and storing their 5-tuple (protocol, source address, source port, destination address, destination port).

This enables concurrent use of a single NAT external port for multiple transport sessions, which allows a NAT to successfully process packets in an IP address resource limited network (e.g., deployment with high address space multiplicative factor (refer to Appendix B. of [RFC6269])).

4. Address Pooling Paired (APP)

The "IP address pooling" behavior of "Paired" (APP) was recommended in REQ-2 from [RFC4787], but the behavior when an external IPv4 runs out of ports was left undefined.

Clarification: This document clarifies that if APP is enabled, new sessions from a host that already has a mapping associated with an external IP that ran out of ports SHOULD be dropped. A configuration parameter MAY be provided to allow a NAT to start using ports from another external IP address when the one that anchored the APP mapping ran out of ports. Tweaking this configuration parameter is a trade-off between service continuity and APP strict enforcement. Note, this behavior is sometimes referred to as 'soft-APP'.

As a reminder, the recommendation for the particular case of a CGN is that an implementation must use the same external IP address mapping for all sessions associated with the same internal IP address, be they TCP, UDP, ICMP, something else, or a mix of different protocols [RFC6888].

Update: This behavior SHOULD apply also for TCP.

5. Endpoint-Independent Mapping (EIM) Protocol Independence

REQ-1 from [RFC4787] and REQ-1 from [RFC5382] do not specify whether EIM are protocol-dependent or protocol-independent. For example, if an outbound TCP SYN creates a mapping, it is left undefined whether outbound UDP packets can reuse such mapping.

Update: EIM mappings SHOULD be protocol-dependent. A configuration parameter MAY be provided to allow protocols that multiplex TCP and UDP over the same source IP address and port number to use a single mapping. The default value of this configuration parameter MUST be protocol-dependent EIM.

This update is consistent with the stateful NAT64 [RFC6146] that clearly specifies three binding information bases (TCP, UDP, ICMP).

6. Endpoint-Independent Filtering (EIF) Protocol Independence

REQ-8 from [RFC4787] and REQ-3 from [RFC5382] do not specify whether mappings with endpoint-independent filtering (EIF) are protocol-independent or protocol-dependent. For example, if an outbound TCP SYN creates a mapping, it is left undefined whether inbound UDP packets matching that mapping should be accepted or rejected.

Update: EIF filtering SHOULD be protocol-dependent. A configuration parameter MAY be provided to make it protocol-independent. The default value of this configuration parameter MUST be protocol-dependent EIF.

This behavior is aligned with the update in Section 5.

Applications that can be transported over a variety of transport protocols and/or support transport fall back schemes won't experience connectivity failures if the NAT is configured with protocol-independent EIM and protocol-independent EIF.

7. Endpoint-Independent Filtering (EIF) Mapping Refresh

The NAT mapping Refresh direction may have a "NAT Inbound refresh behavior" of "True" according to REQ-6 from [RFC4787], but [RFC4787] does not clarify how this behavior applies to EIF mappings. The issue in question is whether inbound packets that match an EIF mapping but do not create a new session due to a security policy should refresh the mapping timer.

Clarification: This document clarifies that even when a NAT has an inbound refresh behavior set to 'TRUE', such packets SHOULD NOT

refresh the mapping. Otherwise a simple attack of a packet every 2 minutes can keep the mapping indefinitely.

Update: This behavior SHOULD apply also for TCP.

7.1. Outbound Mapping Refresh and Error Packets

Update: In the case of NAT outbound refresh behavior, ICMP Errors or TCP RST outbound packets, sent as response to inbound packets, SHOULD NOT refresh the mapping. Other packets which indicate the host is not interested in receiving packets MAY be configurable to also not refresh state, such as STUN error response [RFC5389] or IKE INVALID_SYNTAX [RFC7296].

8. Port Parity

Update: A NAT MAY disable port parity preservation for all dynamic mappings. Nevertheless, A NAT SHOULD support means to explicitly request to preserve port parity (e.g., [RFC7753]).

Note: According to [RFC6887], dynamic mappings are said to be dynamic in the sense that they are created on demand, either implicitly or explicitly:

1. Implicit dynamic mappings refer to mappings that are created as a side effect of traffic such as an outgoing TCP SYN or outgoing UDP packet. Implicit dynamic mappings usually have a finite lifetime, though this lifetime is generally not known to the client using them.
2. Explicit dynamic mappings refer to mappings that are created as a result, for example, of explicit Port Control Protocol (PCP) MAP and PEER requests. Explicit dynamic mappings have a finite lifetime, and this lifetime is communicated to the client.

9. Port Randomization

Update: A NAT SHOULD follow the recommendations specified in Section 4 of [RFC6056], especially:

"A NAT that does not implement port preservation [RFC4787] [RFC5382] SHOULD obfuscate selection of the ephemeral port of a packet when it is changed during translation of that packet. A NAT that does implement port preservation SHOULD obfuscate the ephemeral port of a packet only if the port must be changed as a result of the port being already in use for some other session. A NAT that performs parity preservation and that

must change the ephemeral port during translation of a packet SHOULD obfuscate the ephemeral ports. The algorithms described in this document could be easily adapted such that the parity is preserved (i.e., force the lowest order bit of the resulting port number to 0 or 1 according to whether even or odd parity is desired)."

10. IP Identification (IP ID)

Update: A NAT SHOULD handle the Identification field of translated IPv4 packets as specified in Section 5.3.1 of [RFC6864].

11. ICMP Query Mappings Timeout

Section 3.1 of [RFC5508] specifies that ICMP Query Mappings are to be maintained by a NAT. However, the specification doesn't discuss Query Mapping timeout values. Section 3.2 of [RFC5508] only discusses ICMP Query Session Timeouts.

Update: ICMP Query Mappings MAY be deleted once the last session using the mapping is deleted.

12. Hairpinning Support for ICMP Packets

REQ-7 from [RFC5508] specifies that a NAT enforcing 'Basic NAT' must support traversal of hairpinned ICMP Query sessions.

Clarification: This implicitly means that address mappings from external address to internal address (similar to Endpoint Independent Filters) must be maintained to allow inbound ICMP Query sessions. If an ICMP Query is received on an external address, a NAT can then translate to an internal IP.

REQ-7 from [RFC5508] specifies that all NATs must support the traversal of hairpinned ICMP Error messages.

Clarification: This behavior requires a NAT to maintain address mappings from external IP address to internal IP address in addition to the ICMP Query Mappings described in Section 3.1 of [RFC5508].

13. IANA Considerations

This document does not require any IANA action.

14. Security Considerations

NAT behavioral considerations are discussed in [RFC4787], [RFC5382], and [RFC5508].

Because some of the clarifications and updates (e.g., Section 2) are inspired from NAT64, the security considerations discussed in Section 5 of [RFC6146] apply also for this specification.

The update in Section 3 allows for an optimized NAT resource usage. In order to avoid service disruption, the NAT must not invoke this functionality unless the packets are to be sent to distinct destination addresses.

Some of the updates (e.g., Section 7, Section 9, and Section 11) allow for an increased security compared to [RFC4787], [RFC5382], and [RFC5508]. Particularly:

- o The updates in Section 7 and Section 11 prevent an illegitimate node to maintain mappings activated in the NAT while these mappings should be cleared.
- o Port randomization (Section 9) complicates tracking hosts located behind a NAT.

Section 4 and Section 12 propose updates that increase the serviceability of a host located behind a NAT. These updates do not introduce any additional security concerns to [RFC4787], [RFC5382], and [RFC5508].

The updates in Section 5 and Section 6 allow for a better NAT transparency from an application standpoint. Hosts that require a restricted filtering behavior should enable specific policies (e.g., access control list (ACL)) either locally or by soliciting a dedicated security device (e.g., firewall). How a host updates its filtering policies is out of scope of this document.

The update in Section 8 induces security concerns that are specific to the protocol used to interact with the NAT. For example, if PCP is used to explicitly request parity preservation for a given mapping, the security considerations discussed in [RFC6887] should be taken into account.

The update in Section 10 may have undesired effects on the performance of the NAT in environments in which fragmentation is massively experienced. Such issue may be used as an attack vector against NATs.

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <<http://www.rfc-editor.org/info/rfc4787>>.
- [RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008, <<http://www.rfc-editor.org/info/rfc5382>>.
- [RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, DOI 10.17487/RFC5508, April 2009, <<http://www.rfc-editor.org/info/rfc5508>>.
- [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <<http://www.rfc-editor.org/info/rfc6056>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<http://www.rfc-editor.org/info/rfc6146>>.
- [RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field", RFC 6864, DOI 10.17487/RFC6864, February 2013, <<http://www.rfc-editor.org/info/rfc6864>>.

15.2. Informative References

- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, DOI 10.17487/RFC2663, August 1999, <<http://www.rfc-editor.org/info/rfc2663>>.

- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <<http://www.rfc-editor.org/info/rfc3022>>.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <<http://www.rfc-editor.org/info/rfc5389>>.
- [RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <<http://www.rfc-editor.org/info/rfc6269>>.
- [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <<http://www.rfc-editor.org/info/rfc6887>>.
- [RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888, April 2013, <<http://www.rfc-editor.org/info/rfc6888>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<http://www.rfc-editor.org/info/rfc7296>>.
- [RFC7753] Sun, Q., Boucadair, M., Sivakumar, S., Zhou, C., Tsou, T., and S. Perreault, "Port Control Protocol (PCP) Extension for Port-Set Allocation", RFC 7753, DOI 10.17487/RFC7753, February 2016, <<http://www.rfc-editor.org/info/rfc7753>>.

Acknowledgements

Thanks to Dan Wing, Suresh Kumar, Mayuresh Bakshi, Rajesh Mohan, Lars Eggert, Gorrry Fairhurst, Brandon Williams, and David Black for their review and discussion.

Many thanks to Ben Laurie for the secdir review, and Dan Romascanu for the Gen-ART review.

Dan Wing proposed some text for the configurable errors in Section 7.1.

Contributors

The following individual contributed text to the document:

Sarat Kamiset, Insieme Networks, United States

Authors' Addresses

Reinaldo Penno
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: repenno@cisco.com

Simon Perreault
Jive Communications
Canada

Email: sperreault@jive.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Senthil Sivakumar
Cisco Systems, Inc.
United States

Email: ssenthil@cisco.com

Kengo Naito
NTT
Tokyo
Japan

Email: k.naito@nttv6.jp

TSVWG Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: October 6, 2016

G. Fairhurst
University of Aberdeen
April 04, 2016

Network Transport Circuit Breakers
draft-ietf-tsvwg-circuit-breaker-15

Abstract

This document explains what is meant by the term "network transport Circuit Breaker" (CB). It describes the need for circuit breakers for network tunnels and applications when using non-congestion-controlled traffic, and explains where circuit breakers are, and are not, needed. It also defines requirements for building a circuit breaker and the expected outcomes of using a circuit breaker within the Internet.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 6, 2016.

Copyright Notice

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. Types of Circuit Breaker	6
2. Terminology	6
3. Design of a Circuit-Breaker (What makes a good circuit breaker?)	6
3.1. Functional Components	7
3.2. Other network topologies	10
3.2.1. Use with a multicast control/routing protocol	10
3.2.2. Use with control protocols supporting pre-provisioned capacity	11
3.2.3. Unidirectional Circuit Breakers over Controlled Paths	12
4. Requirements for a Network Transport Circuit Breaker	12
5. Examples of Circuit Breakers	15
5.1. A Fast-Trip Circuit Breaker	15
5.1.1. A Fast-Trip Circuit Breaker for RTP	16
5.2. A Slow-trip Circuit Breaker	17
5.3. A Managed Circuit Breaker	17
5.3.1. A Managed Circuit Breaker for SAToP Pseudo-Wires	17
5.3.2. A Managed Circuit Breaker for Pseudowires (PWs)	18
6. Examples where circuit breakers may not be needed.	19
6.1. CBs over pre-provisioned Capacity	19
6.2. CBs with tunnels carrying Congestion-Controlled Traffic	19
6.3. CBs with Uni-directional Traffic and no Control Path	20
7. Security Considerations	20
8. IANA Considerations	22
9. Acknowledgments	22
10. Revision Notes	22
11. References	25
11.1. Normative References	25
11.2. Informative References	25
Author's Address	27

1. Introduction

The term "Circuit Breaker" originates in electricity supply, and has nothing to do with network circuits or virtual circuits. In electricity supply, a Circuit Breaker is intended as a protection mechanism of last resort. Under normal circumstances, a Circuit Breaker ought not to be triggered; it is designed to protect the supply network and attached equipment when there is overload. People do not expect an electrical circuit-breaker (or fuse) in their home to be triggered, except when there is a wiring fault or a problem with an electrical appliance.

In networking, the Circuit Breaker (CB) principle can be used as a protection mechanism of last resort to avoid persistent excessive congestion impacting other flows that share network capacity. Persistent congestion was a feature of the early Internet of the 1980s. This resulted in excess traffic starving other connections from access to the Internet. It was countered by the requirement to use congestion control (CC) in the Transmission Control Protocol (TCP) [Jacobsen88]. These mechanisms operate in Internet hosts to cause TCP connections to "back off" during congestion. The addition of a congestion control to TCP (currently documented in [RFC5681]) ensured the stability of the Internet, because it was able to detect congestion and promptly react. This was effective in an Internet where most TCP flows were long-lived (ensuring that they could detect and respond to congestion before the flows terminated). Although TCP was by far the dominant traffic, this is no longer the always the case, and non-congestion-controlled traffic, including many applications using the User Datagram Protocol (UDP), can form a significant proportion of the total traffic traversing a link. The current Internet therefore requires that non-congestion-controlled traffic is considered to avoid persistent excessive congestion.

A network transport Circuit Breaker is an automatic mechanism that is used to continuously monitor a flow or aggregate set of flows. The mechanism seeks to detect when the flow(s) experience persistent excessive congestion. When this is detected, a Circuit Breaker terminates (or significantly reduce the rate of) the flow(s). This is a safety measure to prevent starvation of network resources denying other flows from access to the Internet. Such measures are essential for an Internet that is heterogeneous and for traffic that is hard to predict in advance. Avoiding persistent excessive congestion is important to reduce the potential for "Congestion Collapse" [RFC2914].

There are important differences between a transport Circuit Breaker and a congestion control method. Congestion control (as implemented in TCP, SCTP, and DCCP) operates on a timescale on the order of a packet round-trip-time (RTT), the time from sender to destination and return. Congestion at a network link can also be detected using Explicit Congestion Notification (ECN) [RFC3168], which allows the network to signal congestion by marking ECN-capable packets with a Congestion Experienced (CE) mark. Both loss and reception of CE-marked packets are treated as congestion events. Congestion control methods are able to react to a congestion event by continuously adapting to reduce their transmission rate. The goal is usually to limit the transmission rate to a maximum rate that reflects a fair use of the available capacity across a network path. These methods typically operate on individual traffic flows (e.g., a 5-tuple that includes the IP addresses, protocol, and ports).

In contrast, Circuit Breakers are recommended for non-congestion-controlled Internet flows and for traffic aggregates, e.g., traffic sent using a network tunnel. They operate on timescales much longer than the packet RTT, and trigger under situations of abnormal (excessive) congestion. People have been implementing what this document characterizes as circuit breakers on an ad hoc basis to protect Internet traffic. This document therefore provides guidance on how to deploy and use these mechanisms. Later sections provide examples of cases where circuit-breakers may or may not be desirable.

A Circuit Breaker needs to measure (meter) some portion of the traffic to determine if the network is experiencing congestion and needs to be designed to trigger robustly when there is persistent excessive congestion.

A Circuit Breaker trigger will often utilize a series of successive sample measurements metered at an ingress point and an egress point (either of which could be a transport endpoint). The trigger needs to operate on a timescale much longer than the path round trip time (e.g., seconds to possibly many tens of seconds). This longer period is needed to provide sufficient time for transport congestion control (or applications) to adjust their rate following congestion, and for the network load to stabilize after any adjustment. Congestion events can be common when a congestion-controlled transport is used over a network link operating near capacity. Each event results in reduction in the rate of the transport flow experiencing congestion. The longer period seeks to ensure that a Circuit Breaker does not accidentally trigger following a single (or even successive) congestion events.

Once triggered, the Circuit Breaker needs to provide a control function (called the "reaction"). This removes traffic from the network, either by disabling the flow or by significantly reducing the level of traffic. This reaction provides the required protection to prevent persistent excessive congestion being experienced by other flows that share the congested part of the network path.

Section 4 defines requirements for building a Circuit Breaker.

The operational conditions that cause a Circuit Breaker to trigger ought to be regarded as abnormal. Examples of situations that could trigger a Circuit Breaker include:

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

- o traffic generated by an application at a time when the provisioned network capacity is being utilised for other purposes;
- o routing changes that cause additional traffic to start using the path monitored by the Circuit Breaker;
- o misconfiguration of a service/network device where the capacity available is insufficient to support the current traffic aggregate;
- o misconfiguration of an admission controller or traffic policer that allows more traffic than expected across the path monitored by the Circuit Breaker.

Other mechanisms could also be available to network operators to detect excessive congestion (e.g., an observation of excessive utilisation for a port on a network device). Utilising such information, operational mechanisms could react to reduce network load over a shorter timescale than those of a network transport Circuit Breaker. The role of the Circuit Breaker over such paths remains as a method of last resort. Because it acts over a longer timescale, the Circuit Breaker ought to trigger only when other reactions did not succeed in reducing persistent excessive congestion.

In many cases, the reason for triggering a Circuit Breaker will not be evident to the source of the traffic (user, application, endpoint, etc). A Circuit Breaker can be used to limit traffic from applications that are unable, or choose not, to use congestion control, or where the congestion control properties of the traffic cannot be relied upon (e.g., traffic carried over a network tunnel). In such circumstances, it is all but impossible for the Circuit Breaker to signal back to the impacted applications. In some cases applications could therefore have difficulty in determining that a Circuit Breaker has triggered, and where in the network this happened.

Application developers are therefore advised, where possible, to deploy appropriate congestion control mechanisms. An application that uses congestion control will be aware of congestion events in the network. This allows it to regulate the network load under congestion, and is expected to avoid triggering a network Circuit Breaker. For applications that can generate elastic traffic, this will often be a preferred solution.

1.1. Types of Circuit Breaker

There are various forms of network transport circuit breaker. These are differentiated mainly on the timescale over which they are triggered, but also in the intended protection they offer:

- o Fast-Trip Circuit Breakers: The relatively short timescale used by this form of circuit breaker is intended to provide protection for network traffic from a single flow or related group of flows.
- o Slow-Trip Circuit Breakers: This circuit breaker utilizes a longer timescale and is designed to protect network traffic from congestion by traffic aggregates.
- o Managed Circuit Breakers: Utilize the operations and management functions that might be present in a managed service to implement a circuit breaker.

Examples of each type of circuit breaker are provided in section 4.

2. Terminology

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

3. Design of a Circuit-Breaker (What makes a good circuit breaker?)

Although circuit breakers have been talked about in the IETF for many years, there has not yet been guidance on the cases where circuit breakers are needed or upon the design of circuit breaker mechanisms. This document seeks to offer advice on these two topics.

Circuit Breakers are RECOMMENDED for IETF protocols and tunnels that carry non-congestion-controlled Internet flows and for traffic aggregates. This includes traffic sent using a network tunnel. Designers of other protocols and tunnel encapsulations also ought to consider the use of these techniques as a last resort to protect traffic that shares the network path being used.

This document defines the requirements for design of a Circuit Breaker and provides examples of how a Circuit Breaker can be constructed. The specifications of individual protocols and tunnel encapsulations need to detail the protocol mechanisms needed to implement a Circuit Breaker.

Section 3.1 describes the functional components of a circuit breaker and section 3.2 defines requirements for implementing a Circuit Breaker.

3.1. Functional Components

The basic design of a Circuit Breaker involves communication between an ingress point (a sender) and an egress point (a receiver) of a network flow or set of flows. A simple picture of operation is provided in figure 1. This shows a set of routers (each labelled R) connecting a set of endpoints.

A Circuit Breaker is used to control traffic passing through a subset of these routers, acting between the ingress and a egress point network devices. The path between the ingress and egress could be provided by a tunnel or other network-layer technique. One expected use would be at the ingress and egress of a service, where all traffic being considered terminates beyond the egress point, and hence the ingress and egress carry the same set of flows.

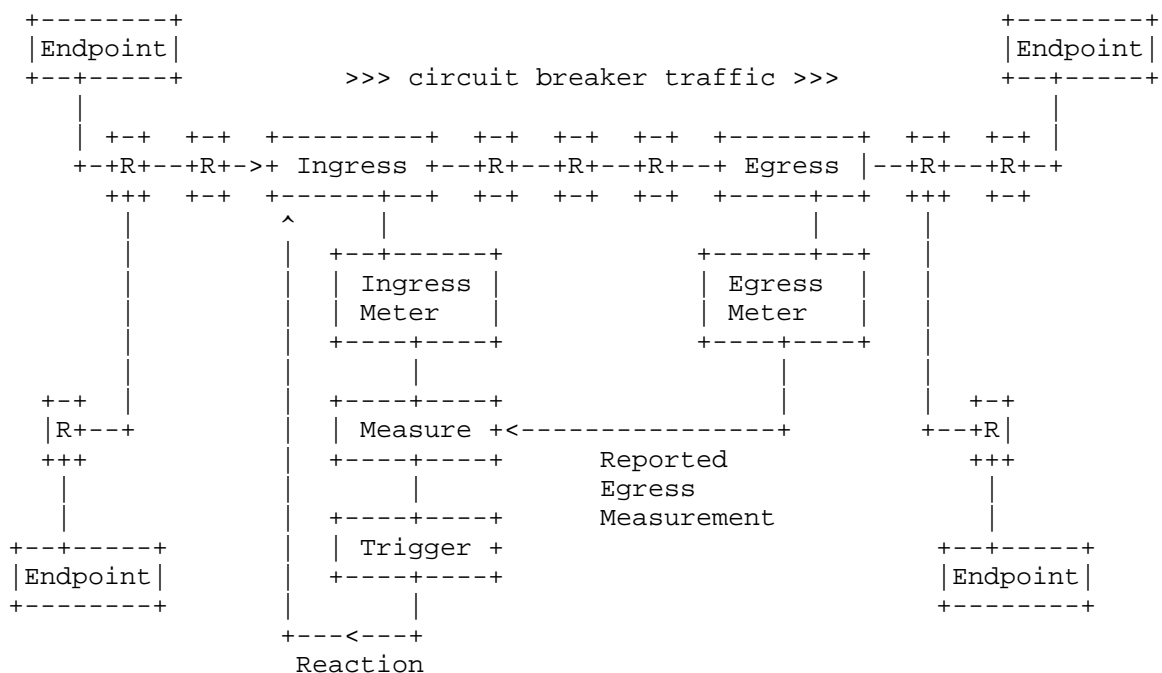


Figure 1: A CB controlling the part of the end-to-end path between an ingress point and an egress point. (Note: In some cases, the trigger

and measurement functions could alternatively be located at other locations (e.g., at a network operations centre.)

In the context of a Circuit Breaker, the ingress and egress functions could be implemented in different places. For example, they could be located in network devices at a tunnel ingress and at the tunnel egress. In some cases, they could be located at one or both network endpoints (see figure 2), implemented as components within a transport protocol.

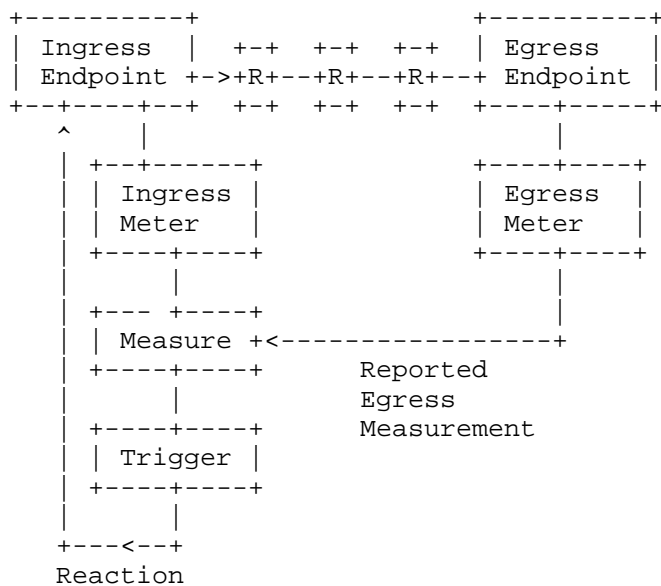


Figure 2: An endpoint CB implemented at the sender (ingress) and receiver (egress).

The set of components needed to implement a Circuit Breaker are:

1. An ingress meter (at the sender or tunnel ingress) that records the number of packets/bytes sent in each measurement interval. This measures the offered network load for a flow or set of flows. For example, the measurement interval could be many seconds (or every few tens of seconds or a series of successive shorter measurements that are combined by the Circuit Breaker Measurement function).
2. An egress meter (at the receiver or tunnel egress) that records the number/bytes received in each measurement interval. This measures the supported load for the flow or set of flows, and

could utilize other signals to detect the effect of congestion (e.g., loss/congestion marking [RFC3168] experienced over the path). The measurements at the egress could be synchronised (including an offset for the time of flight of the data, or referencing the measurements to a particular packet) to ensure any counters refer to the same span of packets.

3. A method that communicates the measured values at the ingress and egress to the Circuit Breaker Measurement function. This could use several methods including: Sending return measurement packets (or control messages) from a receiver to a trigger function at the sender; an implementation using Operations, Administration and Management (OAM); or sending an in-band signalling datagram to the trigger function. This could also be implemented purely as a control plane function, e.g., using a software-defined network controller.
4. A measurement function that combines the ingress and egress measurements to assess the present level of network congestion. (For example, the loss rate for each measurement interval could be deduced from calculating the difference between ingress and egress counter values.) Note the method does not require high accuracy for the period of the measurement interval (or therefore the measured value, since isolated and/or infrequent loss events need to be disregarded.)
5. A trigger function that determines whether the measurements indicate persistent excessive congestion. This function defines an appropriate threshold for determining that there is persistent excessive congestion between the ingress and egress. This preferably considers a rate or ratio, rather than an absolute value (e.g., more than 10% loss, but other methods could also be based on the rate of transmission as well as the loss rate). The Circuit Breaker is triggered when the threshold is exceeded in multiple measurement intervals (e.g., 3 successive measurements). Designs need to be robust so that single or spurious events do not trigger a reaction.
6. A reaction that is applied at the Ingress when the Circuit Breaker is triggered. This seeks to automatically remove the traffic causing persistent excessive congestion.
7. A feedback control mechanism that triggers when either the receive or ingress and egress measurements are not available, since this also could indicate a loss of control packets (also a symptom of heavy congestion or inability to control the load).

In this figure, each endpoint includes a meter that performs a local egress load measurement. An endpoint also extracts the received ingress measurement from the traffic, and compares the ingress and egress measurements to determine if the Circuit Breaker ought to be triggered. This measurement has to be robust to loss (see previous section). If the Circuit Breaker is triggered, it generates a multicast leave message for the egress (e.g., an IGMP or MLD message sent to the last hop router), which causes the upstream router to cease forwarding traffic to the egress endpoint [RFC1112].

Any multicast router that has no active receivers for a particular multicast group will prune traffic for that group, sending a prune message to its upstream router. This starts the process of releasing the capacity used by the traffic and is a standard multicast routing function (e.g., using Protocol Independent Multicast Sparse Mode (PIM-SM) routing protocol [RFC4601]). Each egress operates autonomously, and the Circuit Breaker "reaction" is executed by the multicast control plane (e.g., by PIM) requiring no explicit signalling by the Circuit Breaker along the communication path used for the control messages. Note: there is no direct communication with the Ingress, and hence a triggered Circuit Breaker only controls traffic downstream of the first hop multicast router. It does not stop traffic flowing from the sender to the first hop router; this is common practice for multicast deployment.

The method could also be used with a multicast tunnel or subnetwork (e.g., Section 5.2, Section 5.3), where a meter at the ingress generates additional control messages to carry the measurement data towards the egress where the egress metering is implemented.

3.2.2. Use with control protocols supporting pre-provisioned capacity

Some paths are provisioned using a control protocol, e.g., flows provisioned using the Multi-Protocol Label Switching (MPLS) services, paths provisioned using the resource reservation protocol (RSVP), networks utilizing Software Defined Network (SDN) functions, or admission-controlled Differentiated Services. Figure 1 shows one expected use case, where in this usage a separate device could be used to perform the measurement and trigger functions. The reaction generated by the trigger could take the form of a network control message sent to the ingress and/or other network elements causing these elements to react to the Circuit Breaker. Examples of this type of use are provided in section Section 5.3.

3.2.3. Unidirectional Circuit Breakers over Controlled Paths

A Circuit Breaker can be used to control uni-directional UDP traffic, providing that there is a communication path that can be used for control messages to connect the functional components at the Ingress and Egress. This communication path for the control messages can exist in networks for which the traffic flow is purely unidirectional. For example, a multicast stream that sends packets across an Internet path and can use multicast routing to prune flows to shed network load. Some other types of subnetwork also utilize control protocols that can be used to control traffic flows.

4. Requirements for a Network Transport Circuit Breaker

The requirements for implementing a Circuit Breaker are:

1. There needs to be a communication path for control messages to carry measurement data from the ingress meter and from the egress meter to the point of measurement. (Requirements 16-18 relate to the transmission of control messages.)
2. A CB is REQUIRED to define a measurement period over which the CB Measurement function measures the level of congestion or loss. This method does not have to detect individual packet loss, but MUST have a way to know that packets have been lost/ marked from the traffic flow.
3. An egress meter can also count ECN [RFC3168] congestion marks as a part of measurement of congestion, but in this case, loss MUST also be measured to provide a complete view of the level of congestion. For tunnels, [ID-ietf-tsvwg-tunnel-congestion-feedback] describes a way to measure both loss and ECN-marking; these measurements could be used on a relatively short timescale to drive a congestion control response and/or aggregated over a longer timescale with a higher trigger threshold to drive a CB. Subsequent bullet items in this section discuss the necessity of using a longer timescale and a higher trigger threshold.
4. The measurement period used by a CB Measurement function MUST be longer than the time that current Congestion Control algorithms need to reduce their rate following detection of congestion. This is important because end-to-end Congestion Control algorithms require at least one RTT to notify and adjust the traffic when congestion is experienced, and congestion bottlenecks can share traffic with a diverse range of RTTs. The measurement period is therefore expected to be significantly longer than the RTT experienced by the CB itself.

5. If necessary, a CB MAY combine successive individual meter samples from the ingress and egress to ensure observation of an average measurement over a sufficiently long interval. (Note when meter samples need to be combined, the combination needs to reflect the sum of the individual sample counts divided by the total time/volume over which the samples were measured. Individual samples over different intervals can not be directly combined to generate an average value.)
6. A CB MUST be constructed so that it does not trigger under light or intermittent congestion (see requirements 7-9).
7. A CB is REQUIRED to define a threshold to determine whether the measured congestion is considered excessive.
8. A CB is REQUIRED to define the triggering interval, defining the period over which the trigger uses the collected measurements. CBs need to trigger over a sufficiently long period to avoid additionally penalizing flows with a long path RTT (e.g., many path RTTs).
9. A CB MUST be robust to multiple congestion events. This usually will define a number of measured persistent congestion events per triggering period. For example, a CB MAY combine the results of several measurement periods to determine if the CB is triggered (e.g., it is triggered when persistent excessive congestion is detected in 3 of the measurements within the triggering interval).
10. The normal reaction to a trigger SHOULD disable all traffic that contributed to congestion (otherwise, see requirements 11,12).
11. The reaction MUST be much more severe than that of a Congestion Control algorithm (such as TCP's congestion control [RFC5681] or TCP-Friendly Rate Control, TFRC [RFC5348]), because the CB reacts to more persistent congestion and operates over longer timescales (i.e., the overload condition will have persisted for a longer time before the CB is triggered).
12. A reaction that results in a reduction SHOULD result in reducing the traffic by at least an order of magnitude. A response that achieves the reduction by terminating flows, rather than randomly dropping packets, will often be more desirable to users of the service. A CB that reduces the rate of a flow, MUST continue to monitor the level of congestion and MUST further react to reduce the rate if the CB is again triggered.

13. The reaction to a triggered CB MUST continue for a period that is at least the triggering interval. Operator intervention will usually be required to restore a flow. If an automated response is needed to reset the trigger, then this needs to not be immediate. The design of an automated reset mechanism needs to be sufficiently conservative that it does not adversely interact with other mechanisms (including other CB algorithms that control traffic over a common path). It SHOULD NOT perform an automated reset when there is evidence of continued congestion.
14. A CB trigger SHOULD be regarded as an abnormal network event. As such, this event SHOULD be logged. The measurements that lead to triggering of the CB SHOULD also be logged.
15. The control communication needs to carry measurements (requirement 1) and, in some uses, also needs to transmit trigger messages to the ingress. This control communication may be in-band or out-of-band. The use of in-band communication is RECOMMENDED when either design would be possible. The preferred CB design is one that triggers when it fails to receive measurement reports that indicate an absence of congestion, in contrast to relying on the successful transmission of a "congested" signal back to the sender. (The feedback signal could itself be lost under congestion).

in-Band: An in-band control method SHOULD assume that loss of control messages is an indication of potential congestion on the path, and repeated loss ought to cause the CB to be triggered. This design has the advantage that it provides fate-sharing of the traffic flow(s) and the control communications. This fate-sharing property is weaker when some or all of the measured traffic is sent using a path that differs from the path taken by the control traffic (e.g., where traffic and control messages follow a different path due to use of equal-cost multipath routing, traffic engineering, or tunnels for specific types of traffic).

Out-of-Band: An out-of-band control method SHOULD NOT trigger CB reaction when there is loss of control messages (e.g., a loss of measurements). This avoids failure amplification/propagation when the measurement and data paths fail independently. A failure of an out-of-band communication path SHOULD be regarded as abnormal network event and be handled as appropriate for the network, e.g., this event SHOULD be logged, and additional network operator action might be appropriate, depending on the network and the traffic involved.

16. The control communication MUST be designed to be robust to packet loss. A control message can be lost if there is a failure of the communication path used for the control messages, loss is likely to also be experienced during congestion/overload. This does not imply that it is desirable to provide reliable delivery (e.g., over TCP), since this can incur additional delay in responding to congestion. Appropriate mechanisms could be to duplicate control messages to provide increased robustness to loss, or/and to regard a lack of control traffic as an indication that excessive congestion could be being experienced [ID-ietf-tsvwg-RFC5405.bis]. If control messages traffic are sent over a shared path, it is RECOMMENDED that this control traffic is prioritized to reduce the probability of loss under congestion. Control traffic also needs to be considered when provisioning a network that uses a Circuit Breaker.
17. There are security requirements for the control communication between endpoints and/or network devices (Section 7). The authenticity of the source and integrity of the control messages (measurements and triggers) MUST be protected from off-path attacks. When there is a risk of on-path attack, a cryptographic authentication mechanism for all control/measurement messages is RECOMMENDED.

5. Examples of Circuit Breakers

There are multiple types of Circuit Breaker that could be defined for use in different deployment cases. There could be cases where a flow become controlled by multiple Circuit Breakers (e.g., when the traffic of an end-to-end flow is carried in a tunnel within the network). This section provides examples of different types of Circuit Breaker:

5.1. A Fast-Trip Circuit Breaker

[RFC2309] discusses the dangers of congestion-unresponsive flows and states that "all UDP-based streaming applications should incorporate effective congestion avoidance mechanisms". Some applications do not use a full-featured transport (TCP, SCTP, DCCP). These applications (e.g., using UDP and its UDP-Lite variant) need to provide appropriate congestion avoidance. Guidance for applications that do not use congestion-controlled transports is provided in [ID-ietf-tsvwg-RFC5405.bis]. Such mechanisms can be designed to react on much shorter timescales than a Circuit Breaker, that only observes a traffic envelope. Congestion control methods can also interact with an application to more effectively control its sending rate.

A fast-trip Circuit Breaker is the most responsive form of Circuit Breaker. It has a response time that is only slightly larger than that of the traffic that it controls. It is suited to traffic with well-understood characteristics (and could include one or more trigger functions specifically tailored the type of traffic for which it is designed). It is not suited to arbitrary network traffic and could be unsuitable for traffic aggregates, since it could prematurely trigger (e.g., when the combined traffic from multiple congestion-controlled flows leads to short-term overload).

Although the mechanisms can be implemented in RTP-aware network devices, these mechanisms are also suitable for implementation in endpoints (e.g., as a part of the transport system) where they can also compliment end-to-end congestion control methods. A shorter response time enables these mechanisms to triggers before other forms of Circuit Breaker (e.g., Circuit Breakers operating on traffic aggregates at a point along the network path).

5.1.1. A Fast-Trip Circuit Breaker for RTP

A set of fast-trip Circuit Breaker methods have been specified for use together by a Real-time Transport Protocol (RTP) flow using the RTP/AVP Profile [RTP-CB]. It is expected that, in the absence of severe congestion, all RTP applications running on best-effort IP networks will be able to run without triggering these Circuit Breakers. A fast-trip RTP Circuit Breaker is therefore implemented as a fail-safe that when triggered will terminate RTP traffic.

The sending endpoint monitors reception of in-band RTP Control Protocol (RTCP) reception report blocks, as contained in SR or RR packets, that convey reception quality feedback information. This is used to measure (congestion) loss, possibly in combination with ECN [RFC6679].

The Circuit Breaker action (shutdown of the flow) is triggered when any of the following trigger conditions are true:

1. An RTP Circuit Breaker triggers on reported lack of progress.
2. An RTP Circuit Breaker triggers when no receiver reports messages are received.
3. An RTP Circuit Breaker triggers when the long-term RTP throughput (over many RTTs) exceeds a hard upper limit determined by a method that resembles TCP-Friendly Rate Control (TFRC).
4. An RTP Circuit Breaker includes the notion of Media Usability. This Circuit Breaker is triggered when the quality of the

transported media falls below some required minimum acceptable quality.

5.2. A Slow-trip Circuit Breaker

A slow-trip Circuit Breaker could be implemented in an endpoint or network device. This type of Circuit Breaker is much slower at responding to congestion than a fast-trip Circuit Breaker. This is expected to be more common.

One example where a slow-trip Circuit Breaker is needed is where flows or traffic-aggregates use a tunnel or encapsulation and the flows within the tunnel do not all support TCP-style congestion control (e.g., TCP, SCTP, TFRC), see [ID-ietf-tsvwg-RFC5405.bis] section 3.1.3. A use case is where tunnels are deployed in the general Internet (rather than "controlled environments" within an Internet service provider or enterprise network), especially when the tunnel could need to cross a customer access router.

5.3. A Managed Circuit Breaker

A managed Circuit Breaker is implemented in the signalling protocol or management plane that relates to the traffic aggregate being controlled. This type of Circuit Breaker is typically applicable when the deployment is within a "controlled environment".

A Circuit Breaker requires more than the ability to determine that a network path is forwarding data, or to measure the rate of a path - which are often normal network operational functions. There is an additional need to determine a metric for congestion on the path and to trigger a reaction when a threshold is crossed that indicates persistent excessive congestion.

The control messages can use either in-band or out-of-band communications.

5.3.1. A Managed Circuit Breaker for SAToP Pseudo-Wires

[RFC4553], SAToP Pseudo-Wires (PWE3), section 8 describes an example of a managed Circuit Breaker for isochronous flows.

If such flows were to run over a pre-provisioned (e.g., Multi-Protocol Label Switching, MPLS) infrastructure, then it could be expected that the Pseudowire (PW) would not experience congestion, because a flow is not expected to either increase (or decrease) their rate. If, instead, PW traffic is multiplexed with other traffic over the general Internet, it could experience congestion. [RFC4553] states: "If SAToP PWs run over a PSN providing best-effort service,

they SHOULD monitor packet loss in order to detect "severe congestion". The currently recommended measurement period is 1 second, and the trigger operates when there are more than three measured Severely Errored Seconds (SES) within a period. If such a condition is detected, a SAToP PW ought to shut down bidirectionally for some period of time...".

The concept was that when the packet loss ratio (congestion) level increased above a threshold, the PW was by default disabled. This use case considered fixed-rate transmission, where the PW had no reasonable way to shed load.

The trigger needs to be set at the rate that the PW was likely to experience a serious problem, possibly making the service non-compliant. At this point, triggering the Circuit Breaker would remove the traffic preventing undue impact on congestion-responsive traffic (e.g., TCP). Part of the rationale, was that high loss ratios typically indicated that something was "broken" and ought to have already resulted in operator intervention, and therefore need to trigger this intervention.

An operator-based response to triggering of a Circuit Breaker provides an opportunity for other action to restore the service quality, e.g., by shedding other loads or assigning additional capacity, or to consciously avoid reacting to the trigger while engineering a solution to the problem. This could require the trigger function to send a control message to a third location (e.g., a network operations centre, NOC) that is responsible for operation of the tunnel ingress, rather than the tunnel ingress itself.

5.3.2. A Managed Circuit Breaker for Pseudowires (PWs)

Pseudowires (PWs) [RFC3985] have become a common mechanism for tunneling traffic, and could compete for network resources both with other PWs and with non-PW traffic, such as TCP/IP flows.

[ID-ietf-pals-congcons] discusses congestion conditions that can arise when PWs compete with elastic (i.e., congestion responsive) network traffic (e.g, TCP traffic). Elastic PWs carrying IP traffic (see [RFC4488]) do not raise major concerns because all of the traffic involved responds, reducing the transmission rate when network congestion is detected.

In contrast, inelastic PWs (e.g., a fixed bandwidth Time Division Multiplex, TDM) [RFC4553] [RFC5086] [RFC5087]) have the potential to harm congestion responsive traffic or to contribute to excessive congestion because inelastic PWs do not adjust their transmission rate in response to congestion. [ID-ietf-pals-congcons] analyses TDM

PWs, with an initial conclusion that a TDM PW operating with a degree of loss that could result in congestion-related problems is also operating with a degree of loss that results in an unacceptable TDM service. For that reason, the document suggests that a managed Circuit Breaker that shuts down a PW when it persistently fails to deliver acceptable TDM service is a useful means for addressing these congestion concerns. (See Appendix A of [ID-ietf-pals-congcons] for further discussion.)

6. Examples where circuit breakers may not be needed.

A Circuit Breaker is not required for a single congestion-controlled flow using TCP, SCTP, TFRC, etc. In these cases, the congestion control methods are already designed to prevent persistent excessive congestion.

6.1. CBs over pre-provisioned Capacity

One common question is whether a Circuit Breaker is needed when a tunnel is deployed in a private network with pre-provisioned capacity.

In this case, compliant traffic that does not exceed the provisioned capacity ought not to result in persistent congestion. A Circuit Breaker will hence only be triggered when there is non-compliant traffic. It could be argued that this event ought never to happen - but it could also be argued that the Circuit Breaker equally ought never to be triggered. If a Circuit Breaker were to be implemented, it will provide an appropriate response if persistent congestion occurs in an operational network.

Implementing a Circuit Breaker will not reduce the performance of the flows, but in the event that persistent excessive congestion occurs it protects network traffic that shares network capacity with these flows. It also protects network traffic from a failure when Circuit Breaker traffic is (re)routed to cause additional network load on a non-pre-provisioned path.

6.2. CBs with tunnels carrying Congestion-Controlled Traffic

IP-based traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. A question therefore arises when people deploy a tunnel that is thought to only carry an aggregate of TCP traffic (or traffic using some other congestion control method): Is there advantage in this case in using a Circuit Breaker?

TCP (and SCTP) traffic in a tunnel is expected to reduce the transmission rate when network congestion is detected. Other transports (e.g., using UDP) can employ mechanisms that are sufficient to address congestion on the path [ID-ietf-tsvwg-RFC5405.bis]. However, even if the individual flows sharing a tunnel each implement a congestion control mechanism, and individually reduce their transmission rate when network congestion is detected, the overall traffic resulting from the aggregate of the flows does not necessarily avoid persistent congestion. For instance, most congestion control mechanisms require long-lived flows to react to reduce the rate of a flow. An aggregate of many short flows could result in many flows terminating before they experience congestion. It is also often impossible for a tunnel service provider to know that the tunnel only contains congestion-controlled traffic (e.g., inspecting packet headers might not be possible). Some IP-based applications might not implement adequate mechanisms to address congestion. The important thing to note is that if the aggregate of the traffic does not result in persistent excessive congestion (impacting other flows), then the Circuit Breaker will not trigger. This is the expected case in this context - so implementing a Circuit Breaker ought not to reduce performance of the tunnel, but in the event that persistent excessive congestion occurs the Circuit Breaker protects other network traffic that shares capacity with the tunnel traffic.

6.3. CBs with Uni-directional Traffic and no Control Path

A one-way forwarding path could have no associated communication path for sending control messages, and therefore cannot be controlled using a Circuit Breaker (compare with Section 3.2.3).

A one-way service could be provided using a path with dedicated pre-provisioned capacity that is not shared with other elastic Internet flows (i.e., flows that vary their rate). A forwarding path could also be shared with other flows. One way to mitigate the impact of traffic on the other flows is to manage the traffic envelope by using ingress policing. Supporting this type of traffic in the general Internet requires operator monitoring to detect and respond to persistent excessive congestion.

7. Security Considerations

All Circuit Breaker mechanisms rely upon coordination between the ingress and egress meters and communication with the trigger function. This is usually achieved by passing network control information (or protocol messages) across the network. Timely operation of a Circuit Breaker depends on the choice of measurement period. If the receiver has an interval that is overly long, then

the responsiveness of the Circuit Breaker decreases. This impacts the ability of the Circuit Breaker to detect and react to congestion. If the interval is too short the Circuit Breaker could trigger prematurely resulting in insufficient time for other mechanisms to act, potentially resulting in unnecessary disruption to the service.

A Circuit Breaker could potentially be exploited by an attacker to mount a Denial of Service (DoS) attack against the traffic being controlled by the Circuit Breaker. Mechanisms therefore need to be implemented to prevent attacks on the network control information that would result in DoS.

The authenticity of the source and integrity of the control messages (measurements and triggers) MUST be protected from off-path attacks. Without protection, it could be trivial for an attacker to inject fake or modified control/measurement messages (e.g., indicating high packet loss rates) causing a Circuit Breaker to trigger and to therefore mount a DoS attack that disrupts a flow.

Simple protection can be provided by using a randomized source port, or equivalent field in the packet header (such as the RTP SSRC value and the RTP sequence number) expected not to be known to an off-path attacker. Stronger protection can be achieved using a secure authentication protocol to mitigate this concern.

An attack on the control messages is relatively easy for an attacker on the control path when the messages are neither encrypted nor authenticated. Use of a cryptographic authentication mechanism for all control/measurement messages is RECOMMENDED to mitigate this concern, and would also provide protection from off-path attacks. There is a design trade-off between the cost of introducing cryptographic security for control messages and the desire to protect control communication. For some deployment scenarios the value of additional protection from DoS attack will therefore lead to a requirement to authenticate all control messages.

Transmission of network control messages consumes network capacity. This control traffic needs to be considered in the design of a Circuit Breaker and could potentially add to network congestion. If this traffic is sent over a shared path, it is RECOMMENDED that this control traffic is prioritized to reduce the probability of loss under congestion. Control traffic also needs to be considered when provisioning a network that uses a Circuit Breaker.

The Circuit Breaker MUST be designed to be robust to packet loss that can also be experienced during congestion/overload. Loss of control messages could be a side-effect of a congested network, but also could arise from other causes Section 4.

The security implications depend on the design of the mechanisms, the type of traffic being controlled and the intended deployment scenario. Each design of a Circuit Breaker MUST therefore evaluate whether the particular Circuit Breaker mechanism has new security implications.

8. IANA Considerations

This document makes no request from IANA.

9. Acknowledgments

There are many people who have discussed and described the issues that have motivated this document. Contributions and comments included: Lars Eggert, Colin Perkins, David Black, Matt Mathis, Andrew McGregor, Bob Briscoe and Eliot Lear. This work was part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700).

10. Revision Notes

XXX RFC-Editor: Please remove this section prior to publication XXX

Draft 00

This was the first revision. Help and comments are greatly appreciated.

Draft 01

Contained clarifications and changes in response to received comments, plus addition of diagram and definitions. Comments are welcome.

WG Draft 00

Approved as a WG work item on 28th Aug 2014.

WG Draft 01

Incorporates feedback after Dallas IETF TSVWG meeting. This version is thought ready for WGLC comments. Definitions of abbreviations.

WG Draft 02

Minor fixes for typos. Rewritten security considerations section.

WG Draft 03

Updates following WGLC comments (see TSV mailing list). Comments from C Perkins; D Black and off-list feedback.

A clear recommendation of intended scope.

Changes include: Improvement of language on timescales and minimum measurement period; clearer articulation of endpoint and multicast examples - with new diagrams; separation of the controlled network case; updated text on position of trigger function; corrections to RTP-CB text; clarification of loss v ECN metrics; checks against submission checklist 9use of keywords, added meters to diagrams).

WG Draft 04

Added section on PW CB for TDM - a newly adopted draft (D. Black).

WG Draft 05

Added clarifications requested during AD review.

WG Draft 06

Fixed some remaining typos.

Update following detailed review by Bob Briscoe, and comments by D. Black.

WG Draft 07

Additional update following review by Bob Briscoe.

WG Draft 08

Updated text on the response to lack of meter measurements with managed circuit breakers. Additional comments from Eliot Lear (APPs area).

WG Draft 09

Updated text on applications from Eliot Lear. Additional feedback from Bob Briscoe.

WG Draft 10

Updated text following comments by D Black including a rewritten ECN requirements bullet with of a reference to a tunnel measurement method in [ID-ietf-tsvwg-tunnel-congestion-feedback].

WG Draft 11

Minor corrections after second WGLC.

WG Draft 12

Update following Gen-ART, RTG, and OPS review comments.

WG Draft 13

Fixed a typo.

WG Draft 14

Update after IESG discussion, including:

Reworded introduction. Added definition of ECN.

Requirement

Addressed inconsistency between requirements for control messages. - Removed a "MUST" - following WG feedback on a anearlier version of the draft that "SHOULD" is more appropriate.

Addressed comment about grouping requirements to help show they were inter-related. This reordered some requirements.

Reworded the security considerations.

Corrections to wording to improve clarity.

WG Draft 15 (incorporating pending corrections)

Corrected /applications might be implement/applications might not implement/

Corrected /Inspecting packet headers could/Inspecting packet headers might/

Removed Requirement 9, now duplicated (and renumbered remaining items).

Added "(See Appendix A of [ID-ietf-pals-congcons] for further discussion.)" to end of 5.3.2 - missed comment.

Simplified a sentence in section 6.1, without intended change of meaning.

Added a linking sentence to the second para of Section 6.3.

11. References

11.1. Normative References

- [ID-ietf-tsvwg-RFC5405.bis]
Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines (Work-in-Progress)", 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<http://www.rfc-editor.org/info/rfc3168>>.

11.2. Informative References

- [ID-ietf-pals-congcons]
Stein, YJ., Black, D., and B. Briscoe, "Pseudowire Congestion Considerations (Work-in-Progress)", 2015.
- [ID-ietf-tsvwg-tunnel-congestion-feedback]
Wei, X., Zhu, L., and L. Dend, "Tunnel Congestion Feedback (Work-in-Progress)", 2015.
- [Jacobsen88]
European Telecommunication Standards, Institute (ETSI), "Congestion Avoidance and Control", SIGCOMM Symposium proceedings on Communications architectures and protocols", August 1998.
- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <<http://www.rfc-editor.org/info/rfc1112>>.

- [RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, DOI 10.17487/RFC2309, April 1998, <<http://www.rfc-editor.org/info/rfc2309>>.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<http://www.rfc-editor.org/info/rfc2914>>.
- [RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <<http://www.rfc-editor.org/info/rfc3985>>.
- [RFC4488] Levin, O., "Suppression of Session Initiation Protocol (SIP) REFER Method Implicit Subscription", RFC 4488, DOI 10.17487/RFC4488, May 2006, <<http://www.rfc-editor.org/info/rfc4488>>.
- [RFC4553] Vainshtein, A., Ed. and YJ. Stein, Ed., "Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SATO-P)", RFC 4553, DOI 10.17487/RFC4553, June 2006, <<http://www.rfc-editor.org/info/rfc4553>>.
- [RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <<http://www.rfc-editor.org/info/rfc4601>>.
- [RFC5086] Vainshtein, A., Ed., Sasson, I., Metz, E., Frost, T., and P. Pate, "Structure-Aware Time Division Multiplexed (TDM) Circuit Emulation Service over Packet Switched Network (CESoPSN)", RFC 5086, DOI 10.17487/RFC5086, December 2007, <<http://www.rfc-editor.org/info/rfc5086>>.
- [RFC5087] Stein, Y(J)., Shashoua, R., Insler, R., and M. Anavi, "Time Division Multiplexing over IP (TDMoIP)", RFC 5087, DOI 10.17487/RFC5087, December 2007, <<http://www.rfc-editor.org/info/rfc5087>>.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <<http://www.rfc-editor.org/info/rfc5348>>.

- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<http://www.rfc-editor.org/info/rfc5681>>.
- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <<http://www.rfc-editor.org/info/rfc6679>>.
- [RTP-CB] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions (draft-ietf-avtcore-rtp-circuit-breakers)", February 2014.

Author's Address

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

Email: gorry@erg.abdn.ac.uk
URI: <http://www.erg.abdn.ac.uk>

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).

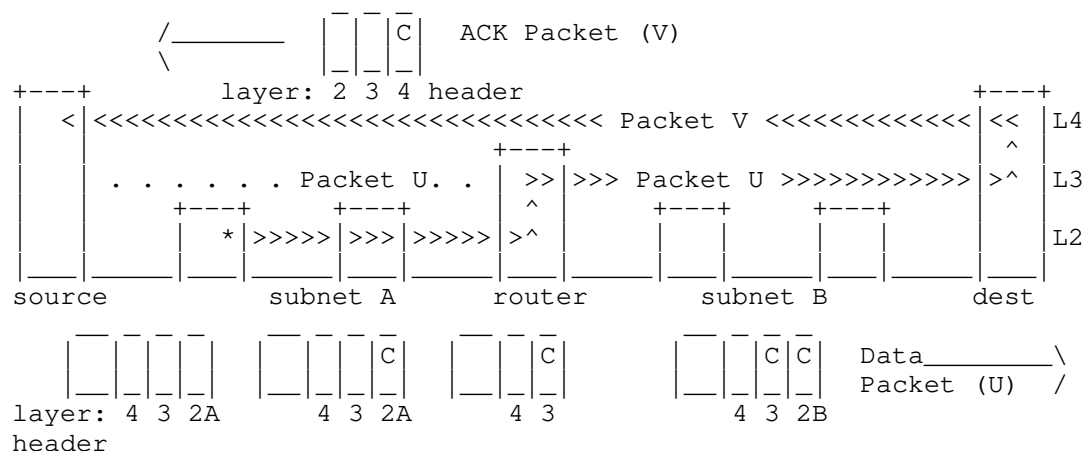


Figure 1: Feed-Forward-and-Up Mode

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.

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.

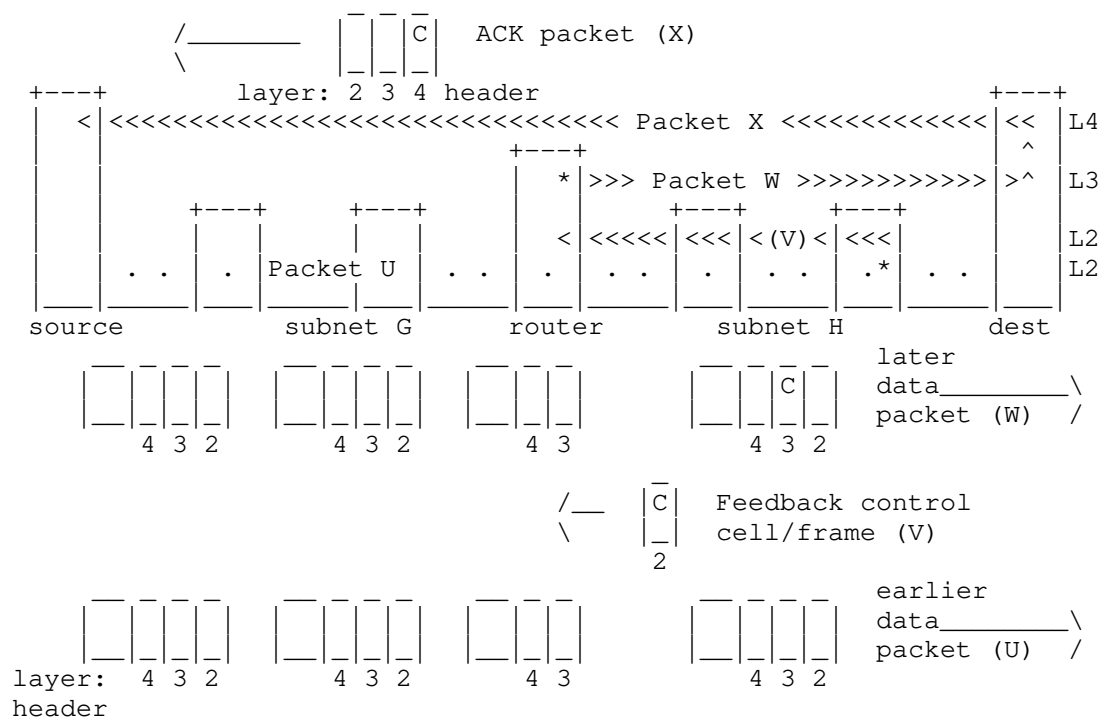


Figure 3: Feed-Backward Mode

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

Network Working Group
Internet-Draft
Intended status: Standard Track

Lucy Yong(Ed.)
Huawei Technologies
E. Crabbe
Oracle
X. Xu
Huawei Technologies
T. Herbert
Facebook

Expires: February 2017

September 30, 2016

GRE-in-UDP Encapsulation
draft-ietf-tsvwg-gre-in-udp-encap-19

Abstract

This document specifies a method of encapsulating network protocol packet within GRE and UDP headers. This GRE-in-UDP encapsulation allows the UDP source port field to be used as an entropy field. This may be used for load balancing of GRE traffic in transit networks using existing ECMP mechanisms. There are two applicability scenarios for GRE-in-UDP with different requirements: (1) general Internet; (2) a traffic-managed controlled environment. The controlled environment has less restrictive requirements than the general Internet.

Status of This Document

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 30, 2017.

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. Terminology.....	4
1.2. Requirements Language.....	5
2. Applicability Statement.....	5
2.1. GRE-in-UDP Tunnel Requirements.....	6
2.1.1. Requirements for Default GRE-in-UDP Tunnel.....	6
2.1.2. Requirements for TMCE GRE-in-UDP Tunnel.....	7
3. GRE-in-UDP Encapsulation.....	7
3.1. IP Header.....	10
3.2. UDP Header.....	10
3.2.1. Source Port.....	10
3.2.2. Destination Port.....	11
3.2.3. Checksum.....	11
3.2.4. Length.....	11
3.3. GRE Header.....	11
4. Encapsulation Process Procedures.....	12
4.1. MTU and Fragmentation.....	12
4.2. Differentiated Services and ECN Marking.....	13
5. Use of DTLS.....	13
6. UDP Checksum Handling.....	14
6.1. UDP Checksum with IPv4.....	14
6.2. UDP Checksum with IPv6.....	14
7. Middlebox Considerations.....	17
7.1. Middlebox Considerations for Zero Checksums.....	18
8. Congestion Considerations.....	18
9. Backward Compatibility.....	19
10. IANA Considerations.....	20
11. Security Considerations.....	21
12. Acknowledgements.....	21
13. Contributors.....	22
14. References.....	23
14.1. Normative References.....	23
14.2. Informative References.....	24
15. Authors' Addresses.....	25

1. Introduction

This document specifies a generic GRE-in-UDP encapsulation for tunneling network protocol packets across an IP network based on Generic Routing Encapsulation (GRE) [RFC2784][RFC7676] and User Datagram Protocol (UDP) [RFC768] headers. The GRE header indicates the payload protocol type via an EtherType [RFC7042] in the protocol type field, and the source port field in the UDP header may be used to provide additional entropy.

A GRE-in-UDP tunnel offers the possibility of better performance for load balancing GRE traffic in transit networks using existing Equal-Cost Multi-Path (ECMP) mechanisms that use a hash of the five-tuple of source IP address, destination IP address, UDP/TCP source port, UDP/TCP destination port. While such hashing distributes UDP and Transmission Control Protocol (TCP) [RFC793] traffic between a common pair of IP addresses across paths, it uses a single path for corresponding GRE traffic because only the two IP addresses and protocol/next header fields participate in the ECMP hash. Encapsulating GRE in UDP enables use of the UDP source port to provide entropy to ECMP hashing.

In addition, GRE-in-UDP enables extending use of GRE across networks that otherwise disallow it; for example, GRE-in-UDP may be used to bridge two islands where GRE is not supported natively across the middleboxes.

GRE-in-UDP encapsulation may be used to encapsulate already tunneled traffic, i.e., tunnel-in-tunnel. In this case, GRE-in-UDP tunnels treat the endpoints of the outer tunnel as the end hosts; the presence of an inner tunnel does not change the outer tunnel's handling of network traffic.

A GRE-in-UDP tunnel is capable of carrying arbitrary traffic and behaves as a UDP application on an IP network. However, a GRE-in-UDP tunnel carrying certain types of traffic does not satisfy the requirements for UDP applications on the Internet [RFC5405bis]. GRE-in-UDP tunnels that do not satisfy these requirements MUST NOT be deployed to carry such traffic over the Internet. For this reason, this document specifies two deployment scenarios for GRE-in-UDP tunnels with GRE-in-UDP tunnel requirements for each of them: (1) general Internet; (2) a traffic-managed controlled environment (TMCE). The TMCE scenario has less restrictive technical requirements for the protocol but more restrictive management and operation requirements for the network by comparison to the general Internet scenario.

To provide security for traffic carried by a GRE-in-UDP tunnel, this document also specifies Datagram Transport Layer Security (DTLS) for GRE-in-UDP tunnels, which SHOULD be used when security is a concern.

GRE-in-UDP encapsulation usage requires no changes to the transit IP network. ECMP hash functions in most existing IP routers may utilize and benefit from the additional entropy enabled by GRE-in-UDP tunnels without any change or upgrade to their ECMP implementation. The encapsulation mechanism is applicable to a variety of IP networks including Data Center and Wide Area Networks, as well as both IPv4 and IPv6 networks.

1.1. Terminology

The terms defined in [RFC768] and [RFC2784] are used in this document. Following are additional terms used in this draft.

Decapsulator: a component performing packet decapsulation at tunnel egress.

ECMP: Equal-Cost Multi-Path.

Encapsulator: a component performing packet encapsulation at tunnel egress.

Flow Entropy: The information to be derived from traffic or applications and to be used by network devices in ECMP process [RFC6438].

Default GRE-in-UDP Tunnel: A GRE-in-UDP tunnel that can apply to the general Internet.

TMCE: A Traffic-managed controlled environment, i.e. an IP network that is traffic-engineered and/or otherwise managed (e.g., via use of traffic rate limiters) to avoid congestion, as defined in Section 2.

TMCE GRE-in-UDP Tunnel: A GRE-in-UDP tunnel that can only apply to a traffic-managed controlled environment that is defined in Section 2.

Tunnel Egress: A tunnel end point that performs packet decapsulation.

Tunnel Ingress: A tunnel end point that performs packet encapsulation.

1.2. Requirements Language

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

2. Applicability Statement

GRE-in-UDP encapsulation applies to IPv4 and IPv6 networks; in both cases, encapsulated packets are treated as UDP datagrams. Therefore, a GRE-in-UDP tunnel needs to meet the UDP usage requirements specified in [RFC5405bis]. These requirements depend on both the delivery network and the nature of the encapsulated traffic. For example, the GRE-in-UDP tunnel protocol does not provide any congestion control functionality beyond that of the encapsulated traffic. Therefore, a GRE-in-UDP tunnel **MUST** be used only with congestion controlled traffic (e.g., IP unicast traffic) and/or within a network that is traffic-managed to avoid congestion.

[RFC5405bis] describes two applicability scenarios for UDP applications: 1) General Internet and 2) A controlled environment. The controlled environment means a single administrative domain or bilaterally agreed connection between domains. A network forming a controlled environment can be managed/operated to meet certain conditions while the general Internet cannot be; thus the requirements for a tunnel protocol operating under a controlled environment can be less restrictive than the requirements in the general Internet.

For the purpose of this document, a traffic-managed controlled environment (TMCE) is defined as an IP network that is traffic-engineered and/or otherwise managed (e.g., via use of traffic rate limiters) to avoid congestion.

This document specifies GRE-in-UDP tunnel usage in the general Internet and GRE-in-UDP tunnel usage in a traffic-managed controlled environment and uses "default GRE-in-UDP tunnel" and "TMCE GRE-in-UDP tunnel" terms to refer to each usage.

NOTE: Although this document specifies two different sets of GRE-in-UDP tunnel requirements based on tunnel usage, the tunnel implementation itself has no ability to detect how and where it is deployed. Therefore it is the responsibility of the user or operator who deploys a GRE-in-UDP tunnel to ensure that it meets the appropriate requirements.

2.1. GRE-in-UDP Tunnel Requirements

This section states out the requirements for a GRE-in-UDP tunnel. Section 2.1.1 describes the requirements for a default GRE-in-UDP tunnel that is suitable for the general Internet; Section 2.1.2 describes a set of relaxed requirements for a TMCE GRE-in-UDP tunnel used in a traffic-managed controlled environment. Both Sections 2.1.1 and 2.1.2 are applicable to an IPv4 or IPv6 delivery network.

2.1.1. Requirements for Default GRE-in-UDP Tunnel

The following is a summary of the default GRE-in-UDP tunnel requirements:

1. A UDP checksum SHOULD be used when encapsulating in IPv4.
2. A UDP checksum MUST be used when encapsulating in IPv6.
3. GRE-in-UDP tunnel MUST NOT be deployed or configured to carry traffic that is not congestion controlled. As stated in [RFC5405bis], IP-based unicast traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. A default GRE-in-UDP tunnel is not appropriate for traffic that is not known to be congestion-controlled (e.g., most IP multicast traffic).
4. UDP source port values that are used as a source of flow entropy SHOULD be chosen from the ephemeral port range (49152-65535) [RFC5405bis].
5. The use of the UDP source port MUST be configurable so that a single value can be set for all traffic within the tunnel (this disables use of the UDP source port to provide flow entropy). When a single value is set, a random port SHOULD be selected in order to minimize the vulnerability to off-path attacks [RFC6056].
6. For IPv6 delivery networks, the flow entropy SHOULD also be placed in the flow label field for ECMP per [RFC6438].
7. At the tunnel ingress, any fragmentation of the incoming packet (e.g., because the tunnel has a Maximum Transmission Unit (MTU) that is smaller than the packet) SHOULD be performed before encapsulation. In addition, the tunnel ingress MUST apply the UDP checksum to all encapsulated fragments so that the tunnel egress can validate reassembly of the fragments; it MUST set the same Differentiated Services Code Point (DSCP) value as in the Differentiated Services

(DS) field of the payload packet in all fragments [RFC2474]. To avoid unwanted forwarding over multiple paths, the same source UDP port value SHOULD be set in all packet fragments.

2.1.2. Requirements for TMCE GRE-in-UDP Tunnel

The section contains the TMCE GRE-in-UDP tunnel requirements. It lists the changed requirements, compared with a Default GRE-in-UDP Tunnel, for a TMCE GRE-in-UDP Tunnel, which corresponds to the requirements 1-3 listed in Section 2.1.1.

1. A UDP checksum SHOULD be used when encapsulating in IPv4. A tunnel endpoint sending GRE-in-UDP MAY disable the UDP checksum, since GRE has been designed to work without a UDP checksum [RFC2784]. However, a checksum also offers protection from mis-delivery to another port.

2. Use of UDP checksum MUST be the default when encapsulating in IPv6. This default MAY be overridden via configuration of UDP zero-checksum mode. All usage of UDP zero-checksum mode with IPv6 is subject to the additional requirements specified in Section 6.2.

3. A GRE-in-UDP tunnel MAY encapsulate traffic that is not congestion controlled.

The requirements 4-7 listed in Section 2.1.1 also apply to a TMCE GRE-in-UDP Tunnel.

3. GRE-in-UDP Encapsulation

The GRE-in-UDP encapsulation format contains a UDP header [RFC768] and a GRE header [RFC2890]. The format is shown as follows: (presented in bit order)

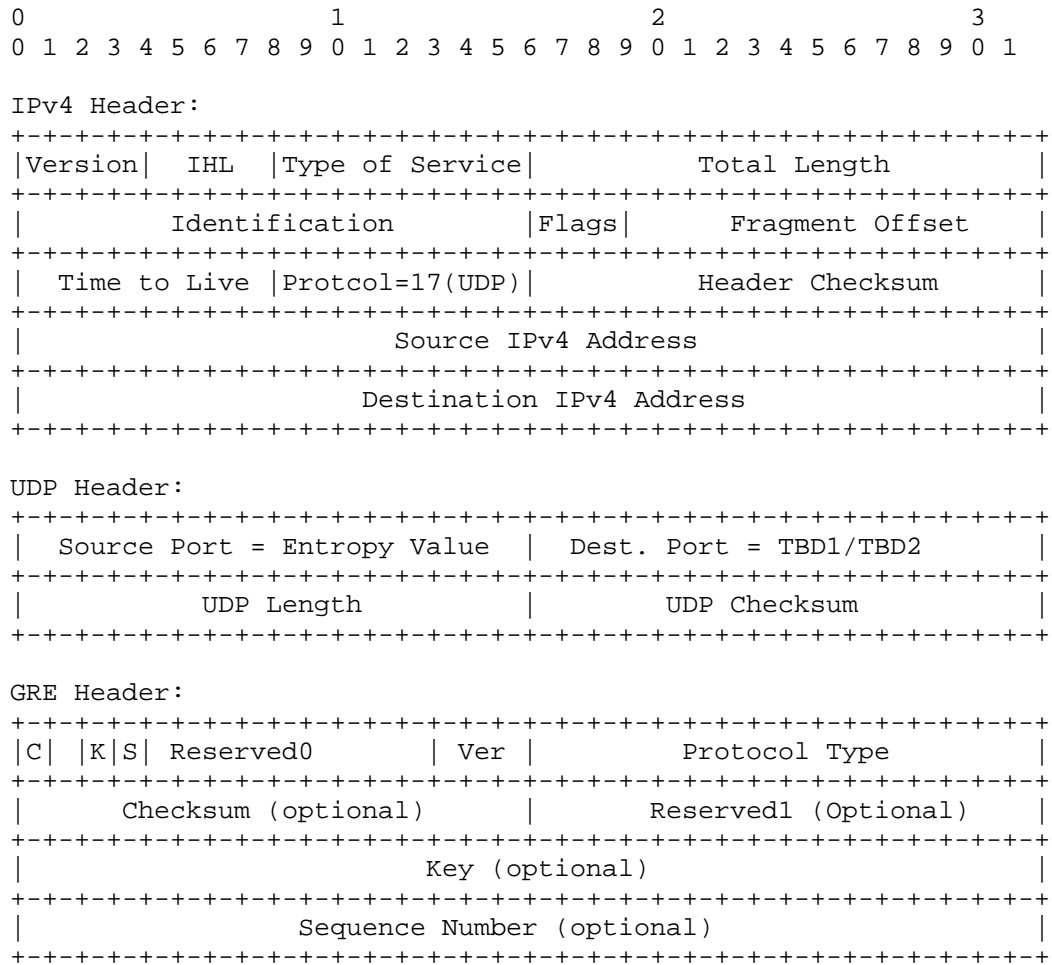


Figure 1 UDP+GRE Headers in IPv4

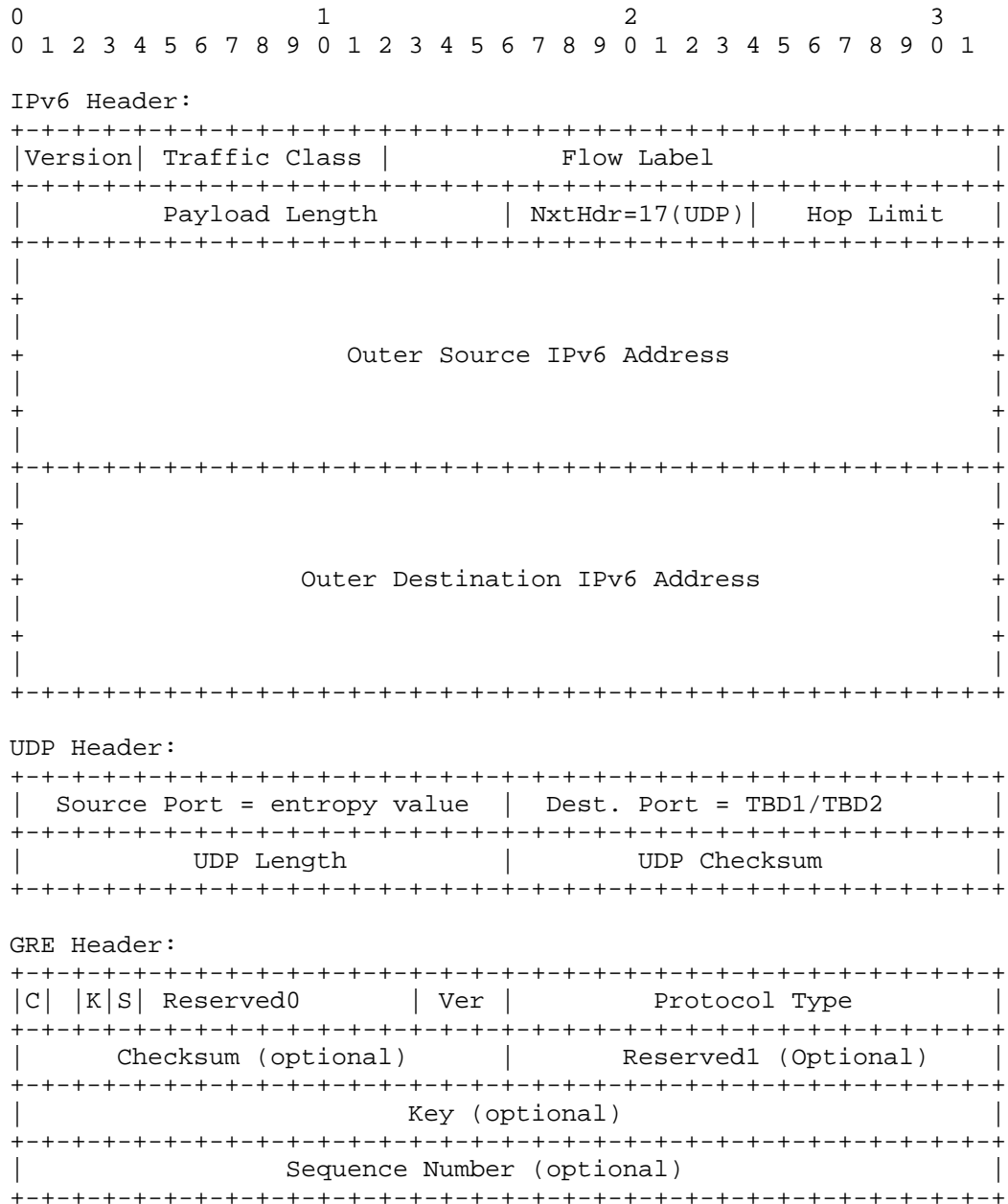


Figure 2 UDP+GRE Headers in IPv6

The contents of the IP, UDP, and GRE headers that are relevant in this encapsulation are described below.

3.1. IP Header

An encapsulator **MUST** encode its own IP address as the source IP address and the decapsulator's IP address as the destination IP address. A sufficiently large value is needed in the IPv4 TTL field or IPv6 Hop Count field to allow delivery of the encapsulated packet to the peer of the encapsulation.

3.2. UDP Header

3.2.1. Source Port

GRE-in-UDP permits the UDP source port value to be used to encode an entropy value. The UDP source port contains a 16-bit entropy value that is generated by the encapsulator to identify a flow for the encapsulated packet. The port value **SHOULD** be within the ephemeral port range, i.e., 49152 to 65535, where the high order two bits of the port are set to one. This provides fourteen bits of entropy for the inner flow identifier. In the case that an encapsulator is unable to derive flow entropy from the payload header or the entropy usage has to be disabled to meet operational requirements (see Section 7), to avoid reordering with a packet flow, the encapsulator **SHOULD** use the same UDP source port value for all packets assigned to a flow e.g., the result of an algorithm that perform a hash of the tunnel ingress and egress IP address.

The source port value for a flow set by an encapsulator **MAY** change over the lifetime of the encapsulated flow. For instance, an encapsulator may change the assignment for Denial of Service (DOS) mitigation or as a means to effect routing through the ECMP network. An encapsulator **SHOULD NOT** change the source port selected for a flow more than once every thirty seconds.

An IPv6 GRE-in-UDP tunnel endpoint **SHOULD** copy a flow entropy value in the IPv6 flow label field (requirement 6). This permits network equipment to inspect this value and utilize it during forwarding, e.g. to perform ECMP [RFC6438].

This document places requirements on the generation of the flow entropy value [RFC5405bis] but does not specify the algorithm that an implementation should use to derive this value.

3.2.2. Destination Port

The destination port of the UDP header is set either GRE-in-UDP (TBD1) or GRE-UDP-DTLS (TBD2) (see Section 5).

3.2.3. Checksum

The UDP checksum is set and processed per [RFC768] and [RFC1122] for IPv4, and [RFC2460] for IPv6. Requirements for checksum handling and use of zero UDP checksums are detailed in Section 6.

3.2.4. Length

The usage of this field is in accordance with the current UDP specification in [RFC768]. This length will include the UDP header (eight bytes), GRE header, and the GRE payload (encapsulated packet).

3.3. GRE Header

An encapsulator sets the protocol type (EtherType) of the packet being encapsulated in the GRE Protocol Type field.

An encapsulator MAY set the GRE Key Present, Sequence Number Present, and Checksum Present bits and associated fields in the GRE header as defined by [RFC2784] and [RFC2890]. Usage of the reserved bits, i.e., Reserved0, is specified in [RFC2784].

The GRE checksum MAY be enabled to protect the GRE header and payload. When the UDP checksum is enabled, it protects the GRE payload, resulting in the GRE checksum being mostly redundant. Enabling both checksums may result in unnecessary processing. Since the UDP checksum covers the pseudo-header and the packet payload, including the GRE header and its payload, the UDP checksum SHOULD be used in preference to using the GRE checksum.

An implementation MAY use the GRE keyid to authenticate the encapsulator. (See Security Considerations Section) In this model, a shared value is either configured or negotiated between an encapsulator and decapsulator. When a decapsulator determines a presented keyid is not valid for the source, the packet MUST be dropped.

Although GRE-in-UDP encapsulation protocol uses both UDP header and GRE header, it is one tunnel encapsulation protocol. GRE and UDP headers MUST be applied and removed as a pair at the encapsulation and decapsulation points. This specification does not support UDP encapsulation of a GRE header where that GRE header is applied or

removed at a network node other than the UDP tunnel ingress or egress.

4. Encapsulation Process Procedures

The procedures specified in this section apply to both a default GRE-in-UDP tunnel and a TMCE GRE-in-UDP tunnel.

The GRE-in-UDP encapsulation allows encapsulated packets to be forwarded through "GRE-in-UDP tunnels". The encapsulator MUST set the UDP and GRE header according to Section 3.

Intermediate routers, upon receiving these UDP encapsulated packets, could load balance these packets based on the hash of the five-tuple of UDP packets.

Upon receiving these UDP encapsulated packets, the decapsulator decapsulates them by removing the UDP and GRE headers and then processes them accordingly.

GRE-in-UDP can encapsulate traffic with unicast, IPv4 broadcast, or multicast (see requirement 3 in Section 2.1.1). However a default GRE-in-UDP tunnel MUST NOT be deployed or configured to carry traffic that is not congestion-controlled (See requirement 3 in Section 2.1.1). Entropy may be generated from the header of encapsulated packets at an encapsulator. The mapping mechanism between the encapsulated multicast traffic and the multicast capability in the IP network is transparent and independent of the encapsulation and is otherwise outside the scope of this document.

To provide entropy for ECMP, GRE-in-UDP does not rely on GRE keep-alive. It is RECOMMENDED not to use GRE keep-alive in the GRE-in-UDP tunnel. This aligns with middlebox traversal guidelines in Section 3.5 of [RFC5405bis].

4.1. MTU and Fragmentation

Regarding packet fragmentation, an encapsulator/decapsulator SHOULD perform fragmentation before the encapsulation. The size of fragments SHOULD be less or equal to the Path MTU (PMTU) associated with the path between the GRE ingress and the GRE egress tunnel endpoints minus the GRE and UDP overhead, assuming the egress MTU for reassembled packets is larger than PMTU. When applying payload fragmentation, the UDP checksum MUST be used so that the receiving endpoint can validate reassembly of the fragments; the same source UDP port SHOULD be used for all packet fragments to ensure the transit routers will forward the fragments on the same path.

If the operator of the transit network supporting the tunnel is able to control the payload MTU size, the MTU SHOULD be configured to avoid fragmentation, i.e., sufficient for the largest supported size of packet, including all additional bytes introduced by the tunnel overhead [RFC5405bis].

4.2. Differentiated Services and ECN Marking

To ensure that tunneled traffic receives the same treatment over the IP network as traffic that is not tunneled, prior to the encapsulation process, an encapsulator processes the tunneled IP packet headers to retrieve appropriate parameters for the encapsulating IP packet header such as DiffServ [RFC2983]. Encapsulation end points that support Explicit Congestion Notification (ECN) must use the method described in [RFC6040] for ECN marking propagation. The congestion control process is outside of the scope of this document.

Additional information on IP header processing is provided in Section 3.1.

5. Use of DTLS

Datagram Transport Layer Security (DTLS) [RFC6347] can be used for application security and can preserve network and transport layer protocol information. Specifically, if DTLS is used to secure the GRE-in-UDP tunnel, the destination port of the UDP header MUST be set to an IANA-assigned value (TBD2) indicating GRE-in-UDP with DTLS, and that UDP port MUST NOT be used for other traffic. The UDP source port field can still be used to add entropy, e.g., for load-sharing purposes. DTLS applies to a default GRE-in-UDP tunnel and a TMCE GRE-in-UDP tunnel.

Use of DTLS is limited to a single DTLS session for any specific tunnel encapsulator/decapsulator pair (identified by source and destination IP addresses). Both IP addresses MUST be unicast addresses - multicast traffic is not supported when DTLS is used. A GRE-in-UDP tunnel decapsulator that supports DTLS is expected to be able to establish DTLS sessions with multiple tunnel encapsulators, and likewise a GRE-in-UDP tunnel encapsulator is expected to be able to establish DTLS sessions with multiple decapsulators. Different source and/or destination IP addresses will be involved (see Section 6.2) for discussion of one situation where use of different source IP addresses is important.

When DTLS is used for a GRE-in-UDP tunnel, if a packet is received from the tunnel and that packet is not protected by the DTLS session

or part of DTLS negotiation (e.g., a DTLS handshake message [RFC6347]), the tunnel receiver MUST discard that packet and SHOULD log that discard event and information about the discarded packet.

DTLS SHOULD be used for a GRE-in-UDP tunnel to meet security requirements of the original traffic that is delivered by a GRE-in-UDP tunnel. There are cases where no additional security is required, e.g., the traffic to be encapsulated is already encrypted or the tunnel is deployed within an operationally secured network. Use of DTLS for a GRE-in-UDP tunnel requires both tunnel endpoints to configure use of DTLS.

6. UDP Checksum Handling

6.1. UDP Checksum with IPv4

For UDP in IPv4, when a non-zero UDP checksum is used, the UDP checksum MUST be processed as specified in [RFC768] and [RFC1122] for both transmit and receive. The IPv4 header includes a checksum that protects against mis-delivery of the packet due to corruption of IP addresses. The UDP checksum potentially provides protection against corruption of the UDP header, GRE header, and GRE payload. Disabling the use of checksums is a deployment consideration that should take into account the risk and effects of packet corruption.

When a decapsulator receives a packet, the UDP checksum field MUST be processed. If the UDP checksum is non-zero, the decapsulator MUST verify the checksum before accepting the packet. By default a decapsulator SHOULD accept UDP packets with a zero checksum. A node MAY be configured to disallow zero checksums per [RFC1122]; this may be done selectively, for instance disallowing zero checksums from certain hosts that are known to be sending over paths subject to packet corruption. If verification of a non-zero checksum fails, a decapsulator lacks the capability to verify a non-zero checksum, or a packet with a zero-checksum was received and the decapsulator is configured to disallow, the packet MUST be dropped and an event MAY be logged.

6.2. UDP Checksum with IPv6

For UDP in IPv6, the UDP checksum MUST be processed as specified in [RFC768] and [RFC2460] for both transmit and receive.

When UDP is used over IPv6, the UDP checksum is relied upon to protect both the IPv6 and UDP headers from corruption. As such, A default GRE-in-UDP Tunnel MUST perform UDP checksum; A TMCE GRE-in-UDP Tunnel MAY be configured with the UDP zero-checksum mode if the

traffic-managed controlled environment or a set of closely cooperating traffic-managed controlled environments (such as by network operators who have agreed to work together in order to jointly provide specific services) meet at least one of following conditions:

- a. It is known (perhaps through knowledge of equipment types and lower layer checks) that packet corruption is exceptionally unlikely and where the operator is willing to take the risk of undetected packet corruption.
- b. It is judged through observational measurements (perhaps of historic or current traffic flows that use a non-zero checksum) that the level of packet corruption is tolerably low and where the operator is willing to take the risk of undetected packet corruption.
- c. Carrying applications that are tolerant of mis-delivered or corrupted packets (perhaps through higher layer checksum, validation, and retransmission or transmission redundancy) where the operator is willing to rely on the applications using the tunnel to survive any corrupt packets.

The following requirements apply to a TMCE GRE-in-UDP tunnel that uses UDP zero-checksum mode:

- a. Use of the UDP checksum with IPv6 MUST be the default configuration of all GRE-in-UDP tunnels.
- b. The GRE-in-UDP tunnel implementation MUST comply with all requirements specified in Section 4 of [RFC6936] and with requirement 1 specified in Section 5 of [RFC6936].
- c. The tunnel decapsulator SHOULD only allow the use of UDP zero-checksum mode for IPv6 on a single received UDP Destination Port regardless of the encapsulator. The motivation for this requirement is possible corruption of the UDP Destination Port, which may cause packet delivery to the wrong UDP port. If that other UDP port requires the UDP checksum, the mis-delivered packet will be discarded.
- d. It is RECOMMENDED that the UDP zero-checksum mode for IPv6 is only enabled for certain selected source addresses. The tunnel decapsulator MUST check that the source and destination IPv6 addresses are valid for the GRE-in-UDP tunnel on which the packet was received if that tunnel uses UDP zero-checksum mode and discard any packet for which this check fails.

- e. The tunnel encapsulator SHOULD use different IPv6 addresses for each GRE-in-UDP tunnel that uses UDP zero-checksum mode regardless of the decapsulator in order to strengthen the decapsulator's check of the IPv6 source address (i.e., the same IPv6 source address SHOULD NOT be used with more than one IPv6 destination address, independent of whether that destination address is a unicast or multicast address). When this is not possible, it is RECOMMENDED to use each source IPv6 address for as few UDP zero-checksum mode GRE-in-UDP tunnels as is feasible.
- f. When any middlebox exists on the path of a GRE-in-UDP tunnel, it is RECOMMENDED to use the default mode, i.e. use UDP checksum, to reduce the chance that the encapsulated packets will be dropped.
- g. Any middlebox that allows the UDP zero-checksum mode for IPv6 MUST comply with requirement 1 and 8-10 in Section 5 of [RFC6936].
- h. Measures SHOULD be taken to prevent IPv6 traffic with zero UDP checksums from "escaping" to the general Internet; see Section 8 for examples of such measures.
- i. IPv6 traffic with zero UDP checksums MUST be actively monitored for errors by the network operator. For example, the operator may monitor Ethernet layer packet error rates.
- j. If a packet with a non-zero checksum is received, the checksum MUST be verified before accepting the packet. This is regardless of whether the tunnel encapsulator and decapsulator have been configured with UDP zero-checksum mode.

The above requirements do not change either the requirements specified in [RFC2460] as modified by [RFC6935] or the requirements specified in [RFC6936].

The requirement to check the source IPv6 address in addition to the destination IPv6 address, plus the strong recommendation against reuse of source IPv6 addresses among GRE-in-UDP tunnels collectively provide some mitigation for the absence of UDP checksum coverage of the IPv6 header. A traffic-managed controlled environment that satisfies at least one of three conditions listed at the beginning of this section provides additional assurance.

A GRE-in-UDP tunnel is suitable for transmission over lower layers in the traffic-managed controlled environments that are allowed by the exceptions stated above and the rate of corruption of the inner

IP packet on such networks is not expected to increase by comparison to GRE traffic that is not encapsulated in UDP. For these reasons, GRE-in-UDP does not provide an additional integrity check except when GRE checksum is used when UDP zero-checksum mode is used with IPv6, and this design is in accordance with requirements 2, 3 and 5 specified in Section 5 of [RFC6936].

Generic Router Encapsulation (GRE) does not accumulate incorrect transport layer state as a consequence of GRE header corruption. A corrupt GRE packet may result in either packet discard or forwarding of the packet without accumulation of GRE state. Active monitoring of GRE-in-UDP traffic for errors is REQUIRED as occurrence of errors will result in some accumulation of error information outside the protocol for operational and management purposes. This design is in accordance with requirement 4 specified in Section 5 of [RFC6936].

The remaining requirements specified in Section 5 of [RFC6936] are not applicable to GRE-in-UDP. Requirements 6 and 7 do not apply because GRE does not include a control feedback mechanism. Requirements 8-10 are middlebox requirements that do not apply to GRE-in-UDP tunnel endpoints (see Section 7.1 for further middlebox discussion).

It is worth mentioning that the use of a zero UDP checksum should present the equivalent risk of undetected packet corruption when sending similar packet using GRE-in-IPv6 without UDP [RFC7676] and without GRE checksums.

In summary, a TMCE GRE-in-UDP Tunnel is allowed to use UDP-zero-checksum mode for IPv6 when the conditions and requirements stated above are met. Otherwise the UDP checksum need to be used for IPv6 as specified in [RFC768] and [RFC2460]. Use of GRE checksum is RECOMMENDED when the UDP checksum is not used.

7. Middlebox Considerations

Many middleboxes read or update UDP port information of the packets that they forward. Network Address/Port Translator (NAPT) is the most commonly deployed Network Address Translation (NAT) device [RFC4787]. An NAPT device establishes a NAT session to translate the {private IP address, private source port number} tuple to a {public IP address, public source port number} tuple, and vice versa, for the duration of the UDP session. This provides a UDP application with the "NAT-pass-through" function. NAPT allows multiple internal hosts to share a single public IP address. The port number, i.e., the UDP Source Port number, is used as the demultiplexer of the multiple internal hosts. However, the above NAPT behaviors conflict

with the behavior a GRE-in-UDP tunnel that is configured to use the UDP source port value to provide entropy.

A GRE-in-UDP tunnel is unidirectional; the tunnel traffic is not expected to be returned back to the UDP source port values used to generate entropy. However some middleboxes (e.g., firewall) assume that bidirectional traffic uses a common pair of UDP ports. This assumption also conflicts with the use of the UDP source port field as entropy.

Hence, use of the UDP source port for entropy may impact middleboxes behavior. If a GRE-in-UDP tunnel is expected to be used on a path with a middlebox, the tunnel can be configured to either disable use of the UDP source port for entropy or to configure middleboxes to pass packets with UDP source port entropy.

7.1. Middlebox Considerations for Zero Checksums

IPv6 datagrams with a zero UDP checksum will not be passed by any middlebox that updates the UDP checksum field or simply validates the checksum based on [RFC2460], such as firewalls. Changing this behavior would require such middleboxes to be updated to correctly handle datagrams with zero UDP checksums. The GRE-in-UDP encapsulation does not provide a mechanism to safely fall back to using a checksum when a path change occurs redirecting a tunnel over a path that includes a middlebox that discards IPv6 datagrams with a zero UDP checksum. In this case the GRE-in-UDP tunnel will be black-holed by that middlebox.

As such, when any middlebox exists on the path of GRE-in-UDP tunnel, use of the UDP checksum is RECOMMENDED to increase the probability of successful transmission of GRE-in-UDP packets. Recommended changes to allow firewalls and other middleboxes to support use of an IPv6 zero UDP checksum are described in Section 5 of [RFC6936].

8. Congestion Considerations

Section 3.1.9 of [RFC5405bis] discusses the congestion considerations for design and use of UDP tunnels; this is important because other flows could share the path with one or more UDP tunnels, necessitating congestion control [RFC2914] to avoid distractive interference.

Congestion has potential impacts both on the rest of the network containing a UDP tunnel, and on the traffic flows using the UDP tunnels. These impacts depend upon what sort of traffic is carried over the tunnel, as well as the path of the tunnel. The GRE-in-UDP

tunnel protocol does not provide any congestion control and GRE-in-UDP packets are regular UDP packets. Therefore, a GRE-in-UDP tunnel MUST NOT be deployed to carry non-congestion controlled traffic over the Internet [RFC5405bis].

Within a TMCE network, GRE-in-UDP tunnels are appropriate for carrying traffic that is not known to be congestion controlled. For example, a GRE-in-UDP tunnel may be used to carry Multiprotocol Label Switching (MPLS) traffic such as pseudowires or VPNs where specific bandwidth guarantees are provided to each pseudowire or VPN. In such cases, operators of TMCE networks avoid congestion by careful provisioning of their networks, rate limiting of user data traffic, and traffic engineering according to path capacity.

When a GRE-in-UDP tunnel carries traffic that is not known to be congestion controlled in a TMCE network, the tunnel MUST be deployed entirely within that network, and measures SHOULD be taken to prevent the GRE-in-UDP traffic from "escaping" the network to the general Internet, e.g.:

- o Physical or logical isolation of the links carrying GRE-in-UDP from the general Internet.
- o Deployment of packet filters that block the UDP ports assigned for GRE-in-UDP.
- o Imposition of restrictions on GRE-in-UDP traffic by software tools used to set up GRE-in-UDP tunnels between specific end systems (as might be used within a single data center) or by tunnel ingress nodes for tunnels that don't terminate at end systems.

9. Backward Compatibility

In general, tunnel ingress routers have to be upgraded in order to support the encapsulations described in this document.

No change is required at transit routers to support forwarding of the encapsulation described in this document.

If a tunnel endpoint (a host or router) that is intended for use as a decapsulator does not support or enable the GRE-in-UDP encapsulation described in this document, that endpoint will not listen on the destination port assigned to the GRE-encapsulation (TBD1 and TBD2). In these cases, the endpoint will perform normal UDP processing and respond to an encapsulator with an ICMP message

indicating "port unreachable" according to [RFC792]. Upon receiving this ICMP message, the node MUST NOT continue to use GRE-in-UDP encapsulation toward this peer without management intervention.

10. IANA Considerations

IANA is requested to make the following allocations:

One UDP destination port number for the indication of GRE,

Service Name: GRE-in-UDP
Transport Protocol(s): UDP
Assignee: IESG <iesg@ietf.org>
Contact: IETF Chair <chair@ietf.org>
Description: GRE-in-UDP Encapsulation
Reference: [This.I-D]
Port Number: TBD1
Service Code: N/A
Known Unauthorized Uses: N/A
Assignment Notes: N/A

Editor Note: replace "TBD1" in section 3 and 9 with IANA assigned number.

One UDP destination port number for the indication of GRE with DTLS,

Service Name: GRE-UDP-DTLS
Transport Protocol(s): UDP
Assignee: IESG <iesg@ietf.org>
Contact: IETF Chair <chair@ietf.org>
Description: GRE-in-UDP Encapsulation with DTLS
Reference: [This.I-D]
Port Number: TBD2
Service Code: N/A
Known Unauthorized Uses: N/A
Assignment Notes: N/A

Editor Note: replace "TBD2" in section 3, 5, and 9 with IANA assigned number.

11. Security Considerations

GRE-in-UDP encapsulation does not affect security for the payload protocol. The security considerations for GRE apply to GRE-in-UDP, see [RFC2784].

To secure traffic carried by a GRE-in-UDP tunnel, DTLS SHOULD be used as specified in Section 5.

In the case that UDP source port for entropy usage is disabled, a random port SHOULD be selected in order to minimize the vulnerability to off-path attacks [RFC6056]. The random port may also be periodically changed to mitigate certain denial of service attacks as mentioned in Section 3.2.1.

Using one standardized value as the UDP destination port to indicate an encapsulation may increase the vulnerability of off-path attack. To overcome this, an alternate port may be agreed upon to use between an encapsulator and decapsulator [RFC6056]. How the encapsulator end points communicate the value is outside scope of this document.

This document does not require that a decapsulator validates the IP source address of the tunneled packets (with the exception that the IPv6 source address MUST be validated when UDP zero-checksum mode is used with IPv6), but it should be understood that failure to do so presupposes that there is effective destination-based (or a combination of source-based and destination-based) filtering at the boundaries.

Corruption of GRE headers can cause security concerns for applications that rely on the GRE key field for traffic separation or segregation. When the GRE key field is used for this purpose such as an application of a Network Virtualization Using Generic Routing Encapsulation (NVGRE) [RFC7637], GRE header corruption is a concern. In such situations, at least one of the UDP and GRE checksums MUST be used for both IPv4 and IPv6 GRE-in-UDP tunnels.

12. Acknowledgements

Authors like to thank Vivek Kumar, Ron Bonica, Joe Touch, Ruediger Geib, Lars Eggert, Lloyd Wood, Bob Briscoe, Rick Casarez, Jouni Korhonen, Kathleen Moriarty, Ben Campbell, and many others for their review and valuable input on this draft.

Thank Donald Eastlake, Eliot Lear, Martin Stiernerling, and Spencer Dawkins for their detail reviews and valuable suggestions in WGLC and IESG process.

Thank the design team led by David Black (members: Ross Callon, Gorrry Fairhurst, Xiaohu Xu, Lucy Yong) to efficiently work out the descriptions for the congestion considerations and IPv6 UDP zero checksum.

Thank David Black and Gorrry Fairhurst for their great help in document content and editing.

13. Contributors

The following people all contributed significantly to this document and are listed below in alphabetical order:

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

Email: david.black@emc.com

Ross Callon
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: rcallon@juniper.net

John E. Drake
Juniper Networks

Email: jdrake@juniper.net

Gorrry Fairhurst
University of Aberdeen

Email: gorrry@erg.abdn.ac.uk

Yongbing Fan
China Telecom
Guangzhou, China.
Phone: +86 20 38639121

Email: fanyb@gsta.com

Adrian Farrel
Juniper Networks

Email: adrian@olddog.co.uk

Vishwas Manral
Hewlett-Packard Corp.
3000 Hanover St, Palo Alto.

Email: vishwas.manral@hp.com

Carlos Pignataro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, NC 27709 USA

Email: cpignata@cisco.com

14. References

14.1. Normative References

- [RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC1122] Braden, R., "Requirements for Internet Hosts -- Communication Layers", RFC1122, October 1989.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC2119, March 1997.
- [RFC2474] Nichols K., Blake S., Baker F., Black D., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", December 1998.

- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.
- [RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC2890, September 2000.
- [RFC5405bis] Eggert, L., "Unicast UDP Usage Guideline for Application Designers", draft-ietf-tsvwg-rfc5405bis, work in progress.
- [RFC6040] Briscoe, B., "Tunneling of Explicit Congestion Notification", RFC6040, November 2010.
- [RFC6347] Rescoria, E., Modadugu, N., "Datagram Transport Layer Security Version 1.2", RFC6347, 2012.
- [RFC6438] Carpenter, B., Amante, S., "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in tunnels", RFC6438, November, 2011.
- [RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, April 2013.
- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.

14.2. Informative References

- [RFC792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.
- [RFC793] DARPA, "Transmission Control Protocol", RFC793, September 1981.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2914] Floyd, S., "Congestion Control Principles", RFC2914, September 2000.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC2983, October 2000.

- [RFC4787] Audet, F., et al, "network Address Translation (NAT) Behavioral Requirements for Unicast UDP", RFC4787, January 2007.
- [RFC6056] Larsen, M. and Gont, F., "Recommendations for Transport-Protocol Port Randomization", RFC6056, January 2011.
- [RFC6438] Carpenter, B., Amante, S., "Using the Ipv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC6438, November 2011.
- [RFC7042] Eastlake 3rd, D. and Abley, J., "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameter", RFC7042, October 2013.
- [RFC7637] Garg, P. and Wang, Y., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC7637, September 2015.
- [RFC7676] Pignataro, C., Bonica, R., Krishnan, S., "IPv6 Support for Generic Routing Encapsulation (GRE)", RFC7676, October 2015.

15. Authors' Addresses

Lucy Yong
Huawei Technologies, USA

Email: lucy.yong@huawei.com

Edward Crabbe
Oracle

Email: edward.crabbe@gmail.com

Xiaohu Xu
Huawei Technologies,
Beijing, China

Email: xuxiaohu@huawei.com

Tom Herbert
Facebook
1 Hacker Way
Menlo Park, CA
Email : tom@herbertland.com

Network 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.

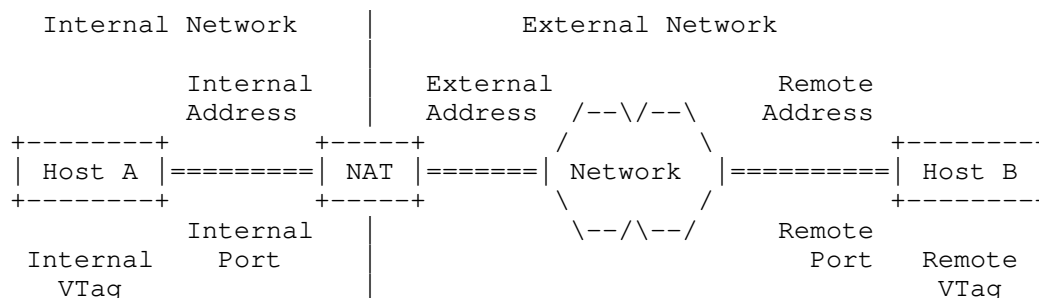


Figure 1: Basic Network Setup

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.

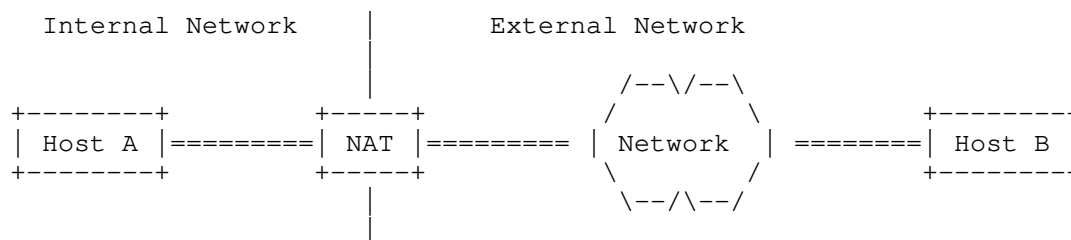


Figure 2: Single NAT Function Scenario

A variation of this case is shown in Figure 3, i.e., multiple NAT functions in the forwarding path between two endpoints.

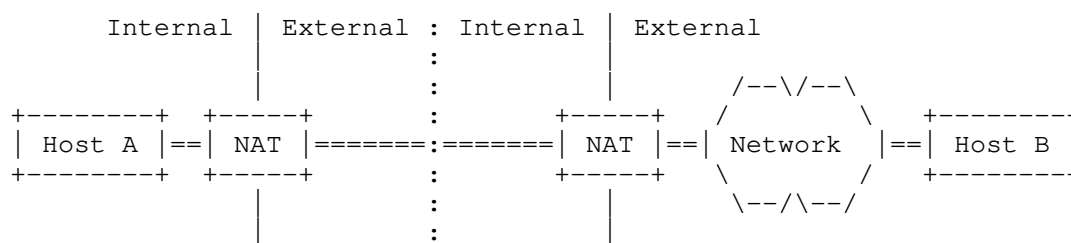


Figure 3: Serial NAT Functions Scenario

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.

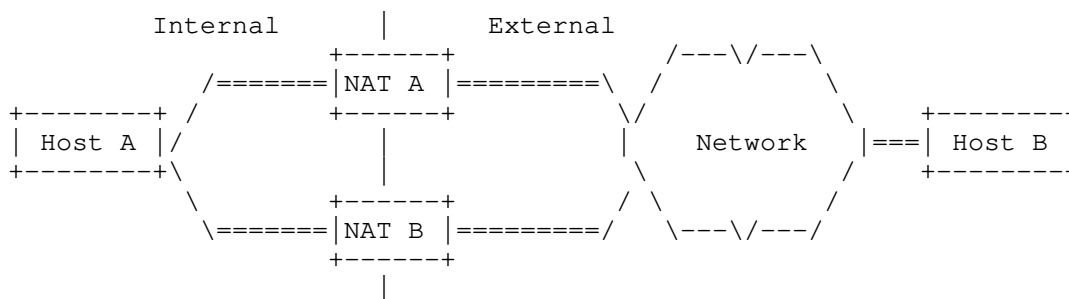


Figure 4: Parallel NAT Functions Scenario

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).

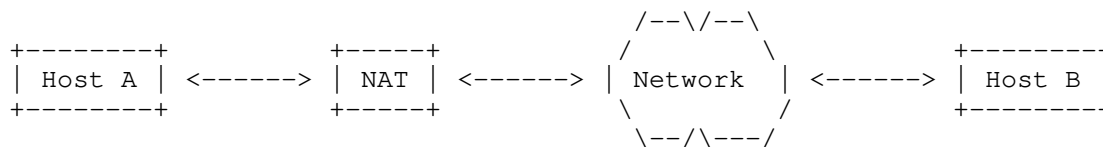
The modification of outgoing SCTP packets sent from an internal host is simple: the source address of the packets has to be replaced with the External-Address. It might also be necessary to establish some state in the NAT function to later handle incoming packets.

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

```

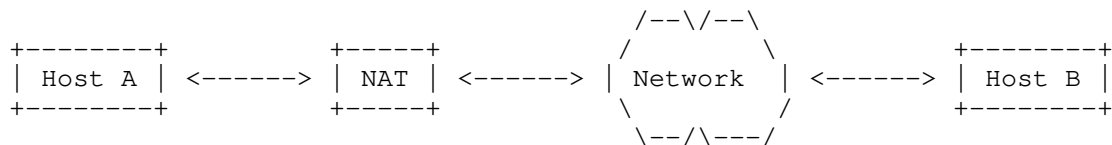
Normally a NAT binding table entry will be created.

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.

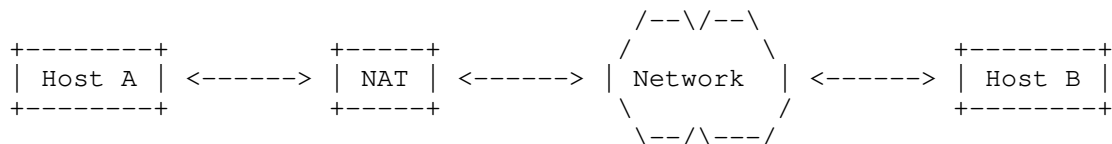


Int-Addr:Int-Port -----> Rem-Addr:Rem-Port
 Rem-VTag

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

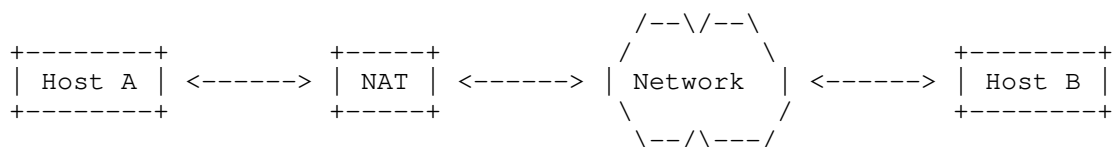
Lookup(Int-VTag, Int-Port, *, Rem-Port)
 Update(*, *, Initiate-Tag, *)

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

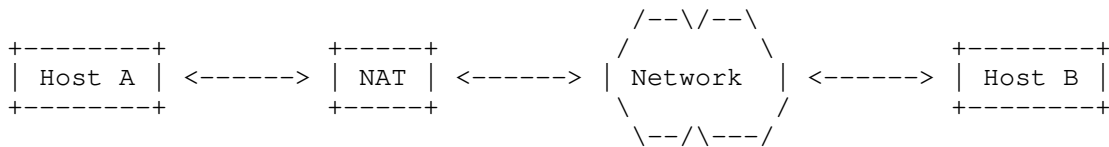
Lookup(*, Int-Port, Rem-VTag, Rem-Port)

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

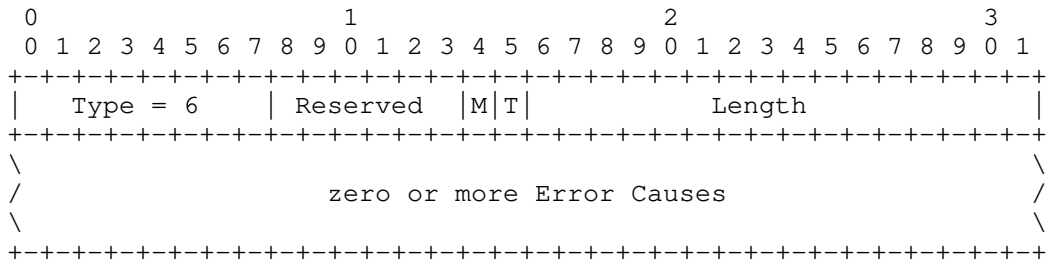
5. Data Formats

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.

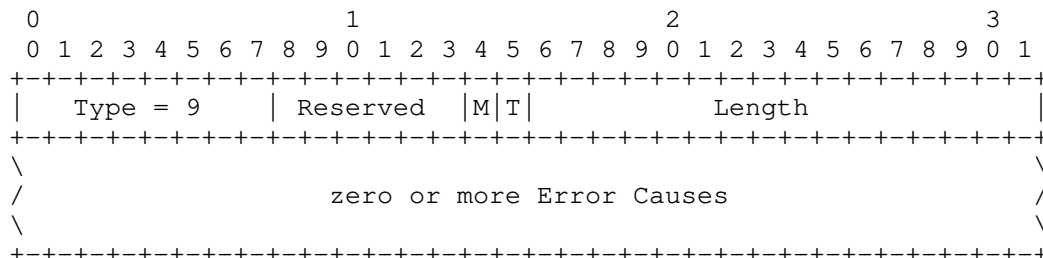
5.1.1. Extended ABORT Chunk



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



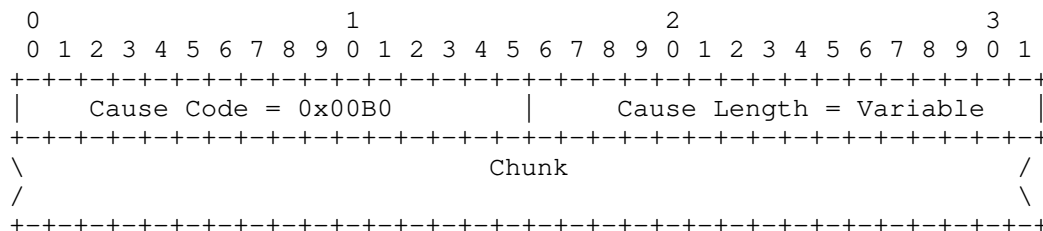
The ERROR chunk defined in [RFC4960] is extended to add the new 'M bit'. The M bit indicates to the receiver of the ERROR chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box.

[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)

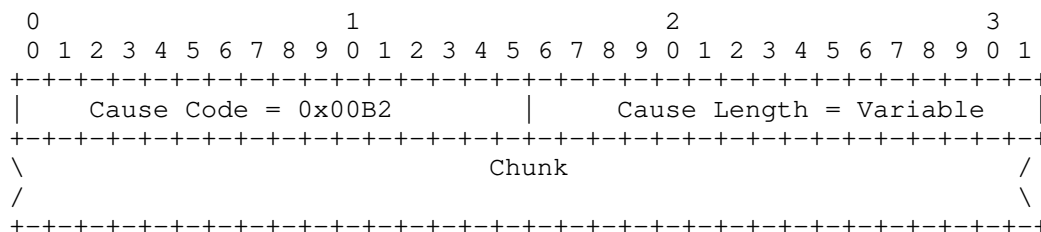
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

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.

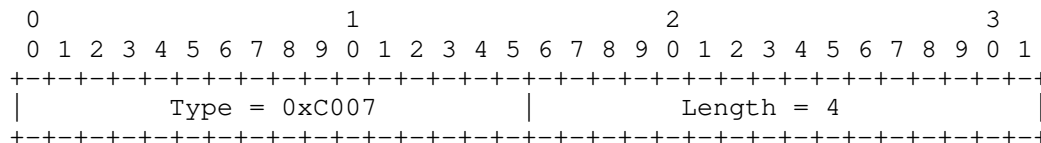
[NOTE to RFC-Editor: Assignment of cause code to be confirmed by IANA.]

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)

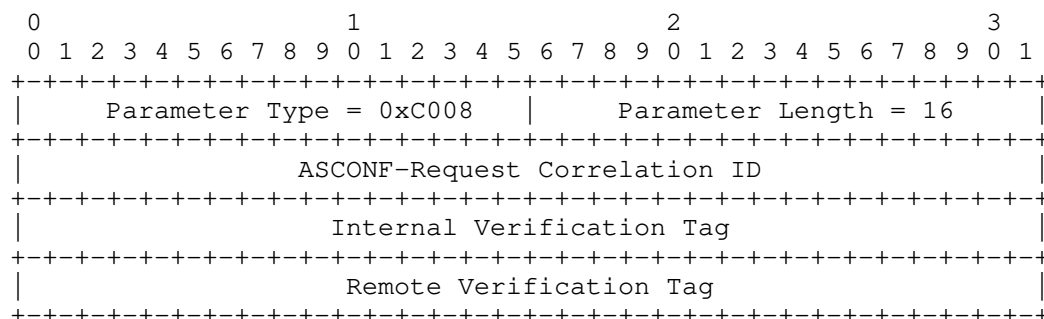
This field holds the length in bytes of the parameter. The value MUST be 4.

[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

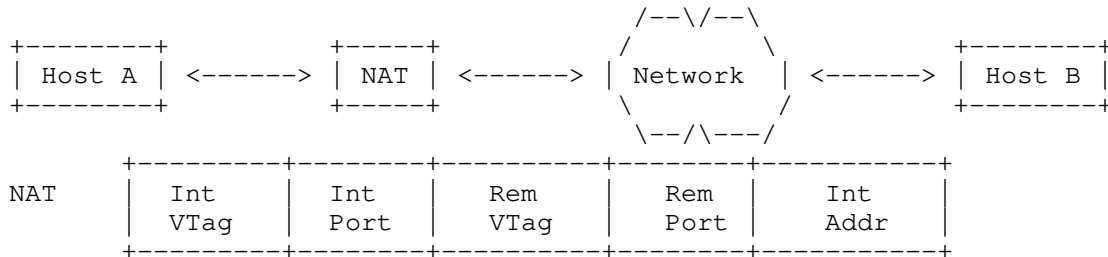
Please note that this section is informational only.

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
```

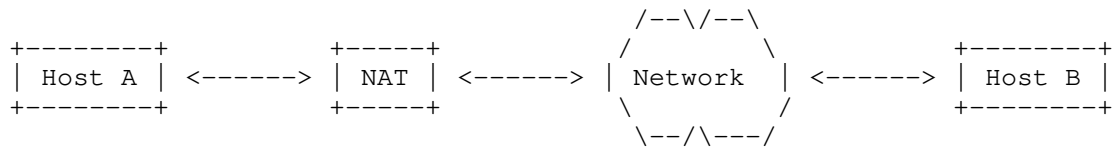
A NAT binding tabled entry is created, the source address is substituted and the packet is sent on:

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.



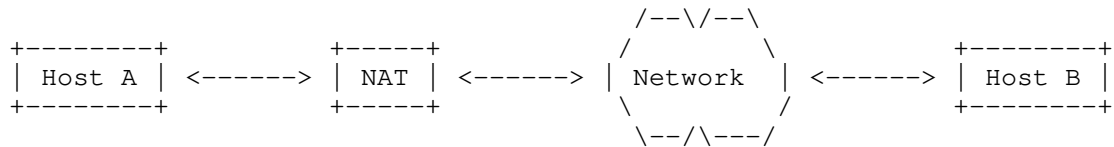
INIT ACK[Initiate-Tag = 5678]
 192.0.2.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

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

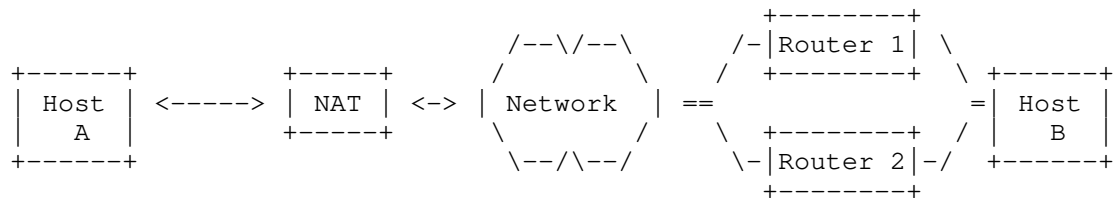
COOKIE ECHO
 192.0.2.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

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
  
```

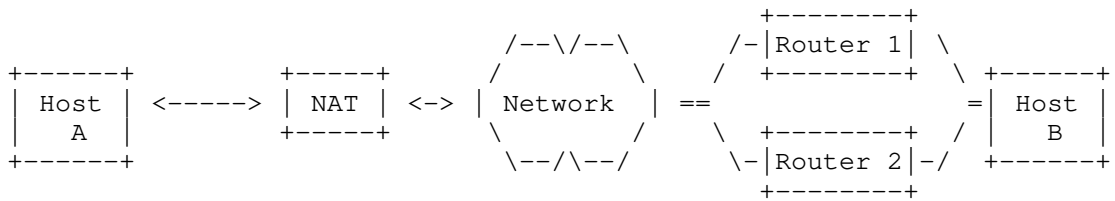
NAT function creates entry:

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.



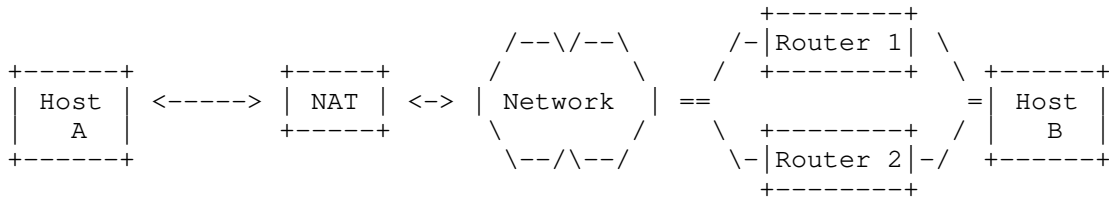
```
INIT ACK[Initiate-tag = 5678, IP-Addr = 203.0.113.129]
192.0.2.1:1 <----- 203.0.113.1:2
                        Int-VTag = 1234
```

The NAT function does not need to change the NAT binding table for the second address:

NAT	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	1234	1	5678	2	10.0.0.1

```
INIT ACK[Initiate-Tag = 5678]
10.0.0.1:1 <--- 203.0.113.1:2
      Int-VTag = 1234
```

The handshake finishes with a COOKIE ECHO acknowledged by a COOKIE ACK.



COOKIE ECHO
10.0.0.1:1 ---> 203.0.113.1:2
Rem-VTag = 5678

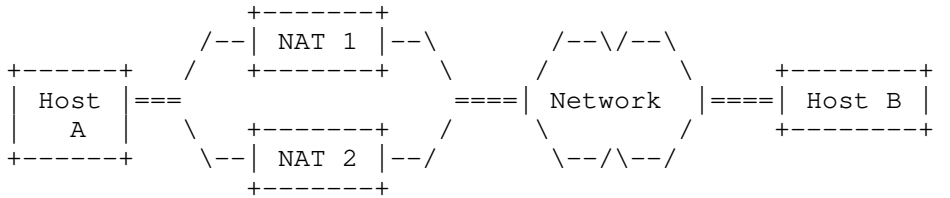
COOKIE ECHO
192.0.2.1:1 -----> 203.0.113.1:2
Rem-VTag = 5678

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:

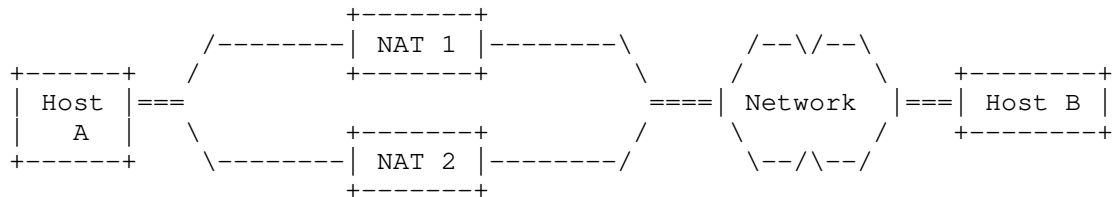
NAT 1	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	0	2	10.0.0.1

```

                                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:

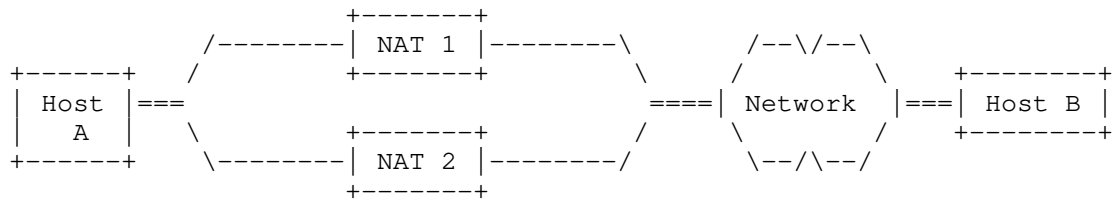
NAT 1	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	5678	2	10.0.0.1

```

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.



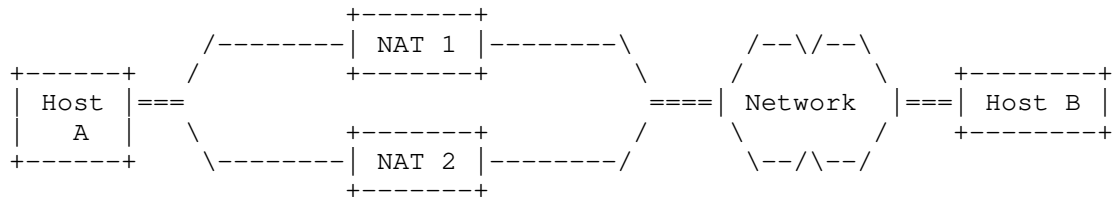
COOKIE ECHO
 10.0.0.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

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

```

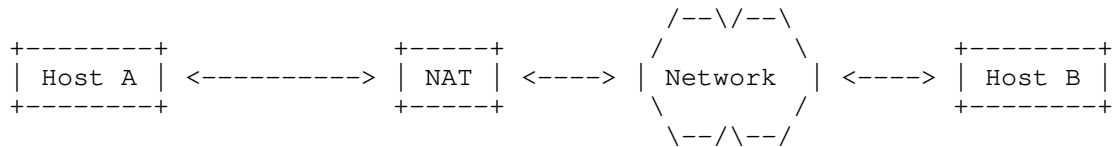
```

                        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.



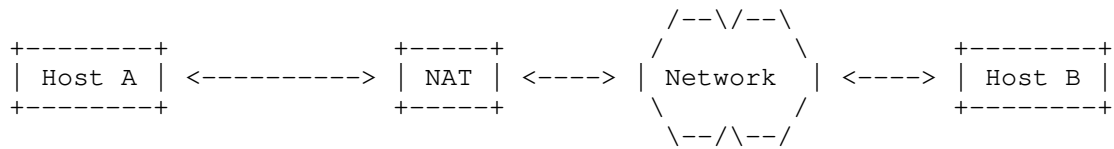
NAT	+-----+				
	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	+-----+				

```

                        DATA
10.0.0.1:1 -----> 203.0.113.1:2
                        Rem-VTag = 5678

```

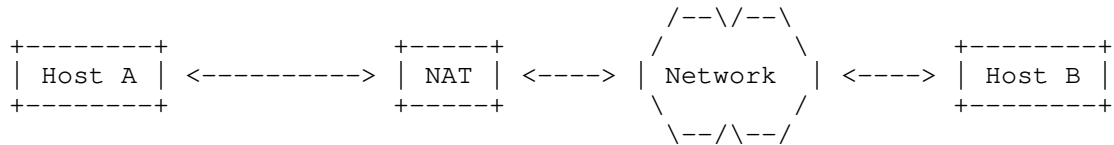
The NAT function cannot find an entry in the NAT binding table for the association. It sends a packet containing an ERROR chunk with the M bit set and the cause "NAT state missing".



```

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.



```

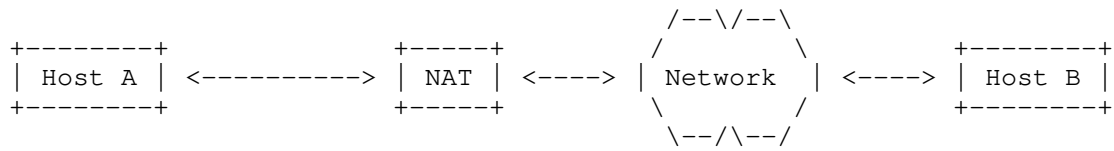
ASCONF [ADD-IP, DELETE-IP, Int-VTag=1234, Rem-VTag = 5678]
10.0.0.1:1 -----> 203.0.113.129:2
      Rem-VTag = 5678
  
```

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

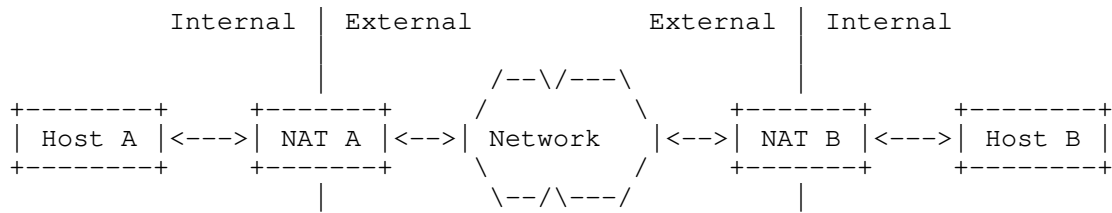
ASCONF ACK
 10.1.0.1:1 <----- 203.0.113.129:2
 Int-VTag = 1234

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

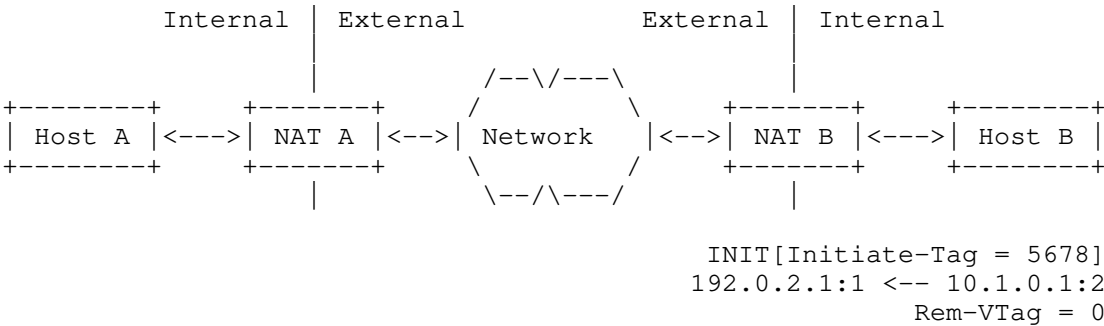
```

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
-------	-------------	-------------	-------------	-------------	-------------

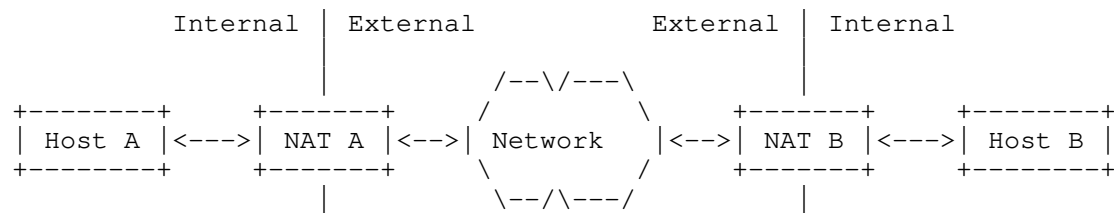
Now Host B sends a packet containing an INIT chunk, which is processed by NAT function B. Its parameters are used to create an entry.



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:

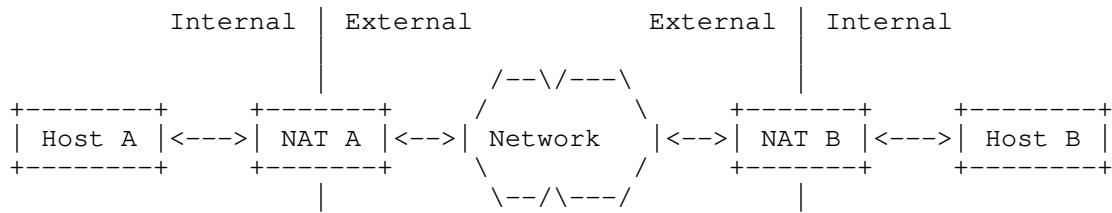


VTag != Int-VTag, but Rem-VTag == 0, find entry.

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:

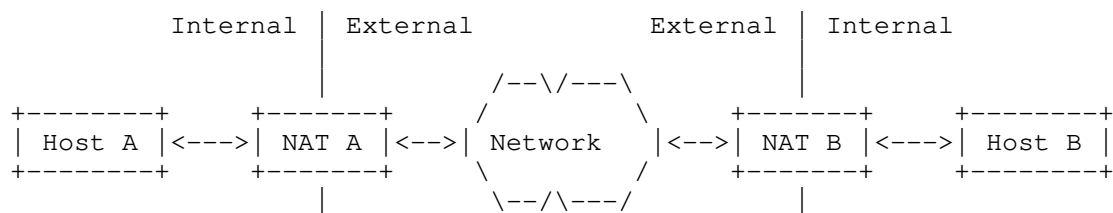
NAT B	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	5678	2	1234	1	10.1.0.1

```

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

TSVWG
Internet Draft
Intended status: Best Current Practice
Expires: October 2015

J. Touch
USC/ISI
April 24, 2015

Recommendations on Using Assigned Transport Port Numbers
draft-ietf-tsvwg-port-use-11.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/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 October 24, 2015.

Copyright Notice

Copyright (c) 2015 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.

Abstract

This document provides recommendations to application and service protocol designers on how to use the assigned transport protocol port number space and when to request a port assignment from IANA. It provides designer guidelines on how to interact with the IANA processes defined in RFC6335, thus serving to complement (but not update) that document.

Table of Contents

1. Introduction.....	2
2. Conventions used in this document.....	3
3. History.....	3
4. Current Port Number Use.....	5
5. What is a Port Number?.....	5
6. Conservation.....	7
6.1. Guiding Principles.....	7
6.2. Firewall and NAT Considerations.....	8
7. Considerations for Requesting Port Number Assignments.....	9
7.1. Is a port number assignment necessary?.....	9
7.2. How Many Assigned Port Numbers?.....	11
7.3. Picking an Assigned Port Number.....	12
7.4. Support for Security.....	13
7.5. Support for Future Versions.....	14
7.6. Transport Protocols.....	15
7.7. When to Request an Assignment.....	16
7.8. Squatting.....	17
7.9. Other Considerations.....	18
8. Security Considerations.....	18
9. IANA Considerations.....	19
10. References.....	19
10.1. Normative References.....	19
10.2. Informative References.....	20
11. Acknowledgments.....	22

1. Introduction

This document provides information and advice to application and service designers on the use of assigned transport port numbers. It provides a detailed historical background of the evolution of transport port numbers and their multiple meanings. It also provides specific recommendations to designers on how to use assigned port numbers. Note that this document provides information to potential port number applicants that complements the IANA process described in BCP165 [RFC6335], but it does not change any of the port number

assignment procedures described therein. This document is intended to address concerns typically raised during Expert Review of assigned port number applications, but it is not intended to bind those reviews. RFC 6335 also describes the interaction between port experts and port requests in IETF consensus document. Authors of IETF consensus documents should nevertheless follow the advice in this document and can expect comment on their port requests from the port experts during IETF last call or at other times when review is explicitly sought.

2. Conventions used in this document

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

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s) indicates a statement using the key words listed above. This convention aids reviewers in quickly identifying or finding requirements for registration and recommendations for use of port numbers in this RFC.

3. History

The term 'port' was first used in [RFC33] to indicate a simplex communication path from an individual process and originally applied to only the Network Control Program (NCP) connection-oriented protocol. At a meeting described in [RFC37], an idea was presented to decouple connections between processes and links that they use as paths, and thus to include numeric source and destination socket identifiers in packets. [RFC38] provides further detail, describing how processes might have more than one of these paths and that more than one path may be active at a time. As a result, there was the need to add a process identifier to the header of each message so that incoming messages could be demultiplexed to the appropriate process. [RFC38] further suggested that 32 bit numbers would be used for these identifiers. [RFC48] discusses the current notion of listening on a specific port number, but does not discuss the issue of port number determination. [RFC61] notes that the challenge of knowing the appropriate port numbers is "left to the processes" in general, but introduces the concept of a "well-known" port number for common services.

[RFC76] proposed a "telephone book" by which an index would allow port numbers to be used by name, but still assumed that both source and destination port numbers are fixed by such a system. [RFC333] proposed that a port number pair, rather than an individual port number, would be used on both sides of the connection for demultiplexing messages. This is the final view in [RFC793] (and its predecessors, including [IEN112]), and brings us to their current meaning. [RFC739] introduced the notion of generic reserved port numbers for groups of protocols, such as "any private RJE server" [RFC739]. Although the overall range of such port numbers was (and remains) 16 bits, only the first 256 (high 8 bits cleared) in the range were considered assigned.

[RFC758] is the first to describe port numbers as being used for TCP (previous RFCs all refer to only NCP). It includes a list of such well-known port numbers, as well as describing ranges used for different purposes:

Decimal	Octal	
---------	-------	--

0-63	0-77	Network Wide Standard Function
64-127	100-177	Hosts Specific Functions
128-223	200-337	Reserved for Future Use
224-255	340-377	Any Experimental Function

In [RFC820] those range meanings disappeared, and a single list of number assignments is presented. This is also the first time that port numbers are described as applying to a connectionless transport (UDP) rather than only connection-oriented transports.

By [RFC900] the ranges appeared as decimal numbers rather than the octal ranges used previously. [RFC1340] increased this range from 0..255 to 0..1023, and began to list TCP and UDP port number assignments individually (although the assumption was that once assigned a port number applies to all transport protocols, including TCP, UDP, recently SCTP and DCCP, as well as ISO-TP4 for a brief period in the early 1990s). [RFC1340] also established the Registered range of 1024-59151, though it notes that it is not controlled by the IANA at that point. The list provided by [RFC1700] in 1994 remained the standard until it was declared replaced by an on-line version, as of [RFC3232] in 2002.

4. Current Port Number Use

RFC6335 indicates three ranges of port number assignments:

Binary	Hex	

0-1023	0x0000-0x03FF	System (also Well-Known)
1024-49151	0x0400-0xBFFF	User (also Registered)
49152-65535	0xC000-0xFFFF	Dynamic (also Private)

System (also Well-Known) encompasses the range 0..1023. On some systems, use of these port numbers requires privileged access, e.g., that the process run as 'root' (i.e., as a privileged user), which is why these are referred to as System port numbers. The port numbers from 1024..49151 denotes non-privileged services, known as User (also Registered), because these port numbers do not run with special privileges. Dynamic (also Private) port numbers are not assigned.

Both System and User port numbers are assigned through IANA, so both are sometimes called 'registered port numbers'. As a result, the term 'registered' is ambiguous, referring either to the entire range 0-49151 or to the User port numbers. Complicating matters further, System port numbers do not always require special (i.e., 'root') privilege. For clarity, the remainder of this document refers to the port number ranges as System, User, and Dynamic, to be consistent with IANA process [RFC6335].

5. What is a Port Number?

A port number is a 16-bit number used for two distinct purposes:

- o Demultiplexing transport endpoint associations within an end host
- o Identifying a service

The first purpose requires that each transport endpoint association (e.g., TCP connection or UDP pairwise association) using a given transport between a given pair of IP addresses use a different pair of port numbers, but does not require either coordination or registration of port number use. It is the second purpose that drives the need for a common registry.

Consider a user wanting to run a web server. That service could run on any port number, provided that all clients knew what port number to use to access that service at that host. Such information can be explicitly distributed - for example, by putting it in the URI:

`http://www.example.com:51509/`

Ultimately, the correlation of a service with a port number is an agreement between just the two endpoints of the association. A web server can run on port number 53, which might appear as DNS traffic to others but will connect to browsers that know to use port number 53 rather than 80.

As a concept, a service is the combination of ISO Layers 5-7 that represents an application protocol capability. For example www (port number 80) is a service that uses HTTP as an application protocol and provides access to a web server [RFC7230]. However, it is possible to use HTTP for other purposes, such as command and control. This is why some current services (HTTP, e.g.) are a bit overloaded - they describe not only the application protocol, but a particular service.

IANA assigns port numbers so that Internet endpoints do not need pairwise, explicit coordination of the meaning of their port numbers. This is the primary reason for requesting port number assignment by IANA - to have a common agreement between all endpoints on the Internet as to the default meaning of a port number, which provides the endpoints with a default port number for a particular protocol or service.

Port numbers are sometimes used by intermediate devices on a network path, either to monitor available services, to monitor traffic (e.g., to indicate the data contents), or to intercept traffic (to block, proxy, relay, aggregate, or otherwise process it). In each case, the intermediate device interprets traffic based on the port number. It is important to recognize that any interpretation of port numbers - except at the endpoints - may be incorrect, because port numbers are meaningful only at the endpoints. Further, port numbers may not be visible to these intermediate devices, such as when the transport protocol is encrypted (as in network- or link-layer tunnels), or when a packet is fragmented (in which case only the first fragment has the port number information). Such port number invisibility may interfere with these in-network port number-based capabilities.

Port numbers can also be used for other purposes. Assigned port numbers can simplify end system configuration, so that individual

installations do not need to coordinate their use of arbitrary port numbers. Such assignments may also have the effect of simplifying firewall management, so that a single, fixed firewall configuration can either permit or deny a service that uses the assigned ports.

It is useful to differentiate a port number from a service name. The former is a numeric value that is used directly in transport protocol headers as a demultiplexing and service identifier. The latter is primarily a user convenience, where the default map between the two is considered static and resolved using a cached index. This document focuses on the former because it is the fundamental network resource. Dynamic maps between the two, i.e., using DNS SRV records, are discussed further in Section 7.1.

6. Conservation

Assigned port numbers are a limited resource that is globally shared by the entire Internet community. As of 2014, approximately 5850 TCP and 5570 UDP port numbers have been assigned out of a total range of 49151. As a result of past conservation, current assigned port use is small and the current rate of assignment avoids the need for transition to larger number spaces. This conservation also helps avoid the need for IANA to rely on assigned port number reclamation, which is practically impossible even though procedurally permitted [RFC6335].

IANA aims to assign only one port number per service, including variants [RFC6335], but there are other benefits to using fewer port numbers for a given service. Use of multiple assigned port numbers can make applications more fragile, especially when firewalls block a subset of those port numbers or use ports numbers to route or prioritize traffic differently. As a result:

>> Each assigned port requested MUST be justified by the applicant as an independently useful service.

6.1. Guiding Principles

This document provides recommendations for users that also help conserve assigned port number space. Again, this document does not update BCP165 [RFC6335], which describes the IANA procedures for managing assigned transport port numbers and services. Assigned port number conservation is based on a number of basic principles:

- o A single assigned port number can support different functions over separate endpoint associations, determined using in-band information. An FTP data connection can transfer binary or text files, the latter translating line-terminators, as indicated in-band over the control port number [RFC959].
- o A single assigned port number can indicate the Dynamic port number(s) on which different capabilities are supported, as with passive-mode FTP [RFC959].
- o Several existing services can indicate the Dynamic port number(s) on which other services are supported, such as with mDNS and portmapper [RFC1833] [RFC6762] [RFC6763].
- o Copies of some existing services can be differentiated using in-band information (e.g., URIs in HTTP Host field and TLS Server Name Indication extension) [RFC7230] [RFC6066].
- o Services requiring varying performance properties can already be supported using separate endpoint associations (connections or other associations), each configured to support the desired properties. E.g., a high-speed and low-speed variant can be determined within the service using the same assigned port.

Assigned port numbers are intended to differentiate services, not variations of performance, replicas, pairwise endpoint associations, or payload types. Assigned port numbers are also a small space compared to other Internet number spaces; it is never appropriate to consume assigned port numbers to conserve larger spaces such as IP addresses, especially where copies of a service represent different endpoints.

6.2. Firewall and NAT Considerations

Ultimately, port numbers indicate services only to the endpoints, and any intermediate device that assigns meaning to a value can be incorrect. End systems might agree to run web services (HTTP) over port number 53 (typically used for DNS) rather than port number 80, at which point a firewall that blocks port number 80 but permits port number 53 would not have the desired effect. Nonetheless, assigned port numbers are often used to help configure firewalls and other port-based systems for access control.

Using Dynamic port numbers, or explicitly-indicated port numbers indicated in-band over another service (such as with FTP) often complicates firewall and NAT interactions [RFC959]. FTP over firewalls often requires direct support for deep-packet inspection

(to snoop for the Dynamic port number for the NAT to correctly map) or passive-mode FTP (in which both connections are opened from the client side).

7. Considerations for Requesting Port Number Assignments

Port numbers are assigned by IANA by a set of documented procedures [RFC6335]. The following section describes the steps users can take to help assist with responsible use of assigned port numbers, and with preparing an application for a port number assignment.

7.1. Is a port number assignment necessary?

First, it is useful to consider whether a port number assignment is required. In many cases, a new number assignment may not be needed, for example:

- o Is this really a new service, or can an existing service suffice?
- o Is this an experimental service [RFC3692]? If so, consider using the current experimental ports [RFC2780].
- o Is this service independently useful? Some systems are composed from collections of different service capabilities, but not all component functions are useful as independent services. Port numbers are typically shared among the smallest independently-useful set of functions. Different service uses or properties can be supported in separate pairwise endpoint associations after an initial negotiation, e.g., to support software decomposition.
- o Can this service use a Dynamic port number that is coordinated out-of-band, e.g.:
 - o By explicit configuration of both endpoints.
 - o By internal mechanisms within the same host (e.g., a configuration file, indicated within a URI, or using interprocess communication).
- o Using information exchanged on a related service: FTP, SIP, etc. [RFC959] [RFC3261].
- o Using an existing port discovery service: portmapper, mDNS, etc. [RFC1833] [RFC6762] [RFC6763].

There are a few good examples of reasons that more directly suggest that not only is a port number assignment not necessary, but it is directly counter-indicated:

- o Assigned port numbers are not intended to differentiate performance variations within the same service, e.g., high-speed vs. ordinary speed. Performance variations can be supported within a single assigned port number in context of separate pairwise endpoint associations.
- o Additional assigned port numbers are not intended to replicate an existing service. For example, if a device is configured to use a typical web browser then it the port number used for that service is a copy of the http service that is already assigned to port number 80 and does not warrant a new assignment. However, an automated system that happens to use HTTP framing - but is not primarily accessed by a browser - might be a new service. A good way to tell is "can an unmodified client of the existing service interact with the proposed service"? If so, that service would be a copy of an existing service and would not merit a new assignment.
- o Assigned port numbers not intended for intra-machine communication. Such communication can already be supported by internal mechanisms (interprocess communication, shared memory, shared files, etc.). When Internet communication within a host is desired, the server can bind to a Dynamic port that is indicated to the client using these internal mechanisms.
- o Separate assigned port numbers are not intended for insecure versions of existing (or new) secure services. A service that already requires security would be made more vulnerable by having the same capability accessible without security.

Note that the converse is different, i.e., it can be useful to create a new, secure service that replicates an existing insecure service on a new port number assignment. This can be necessary when the existing service is not backward-compatible with security enhancements, such as the use of TLS [RFC5246] or DTLS [RFC6347].

- o Assigned port numbers are not intended for indicating different service versions. Version differentiation should be handled in-band, e.g., using a version number at the beginning of an association (e.g., connection or other transaction). This may not be possible with legacy assignments, but all new services should incorporate support for version indication.

Some services may not need assigned port numbers at all, e.g., SIP allows voice calls to use Dynamic ports [RFC3261]. Some systems can register services in the DNS, using SRV entries. These services can be discovered by a variety of means, including mDNS, or via direct query [RFC6762] [RFC6763]. In such cases, users can more easily request a SRV name, which are assigned first-come, first-served from a much larger namespace.

IANA assigns port numbers, but this assignment is typically used only for servers, i.e., the host that listens for incoming connections or other associations. Clients, i.e., hosts that initiate connections or other associations, typically refer to those assigned port numbers but do not need port number assignments for their endpoint.

Finally, an assigned port number is not a guarantee of exclusive use. Traffic for any service might appear on any port number, due to misconfiguration or deliberate misuse. Application and service designers are encouraged to validate traffic based on its content.

7.2. How Many Assigned Port Numbers?

As noted earlier, systems might require a single port number assignment, but rarely require multiple port numbers. There are a variety of known ways to reduce assigned port number consumption. Although some may be cumbersome or inefficient, they are nearly always preferable to consuming additional port number assignments.

Such techniques include:

- o Use of a discovery service, either a shared service (mDNS), or a discovery service for a given system [RFC6762] [RFC6763].
- o Multiplex packet types using in-band information, either on a per-message or per-connection basis. Such demultiplexing can even hand-off different messages and connections among different processes, such as is done with FTP [RFC959].

There are some cases where NAT and firewall traversal are significantly improved by having an assigned port number. Although

NAT traversal protocols supporting automatic configuration have been proposed and developed (e.g., STUN [RFC5389], TURN [RFC5766], and ICE [RFC5245]), not all application and service designers can rely on their presence as of yet.

In the past, some services were assigned multiple port numbers or sometimes fairly large port ranges (e.g., X11). This occurred for a variety of reasons: port number conservation was not as widely appreciated, assignments were not as ardently reviewed, etc. This no longer reflects current practice and such assignments are not considered to constitute a precedent for future assignments.

7.3. Picking an Assigned Port Number

Given a demonstrated need for a port number assignment, the next question is how to pick the desired port number. An application for a port number assignment does not need to include a desired port number; in that case, IANA will select from those currently available.

Users should consider whether the requested port number is important. For example, would an assignment be acceptable if IANA picked the port number value? Would a TCP (or other transport protocol) port number assignment be useful by itself? If so, a port number can be assigned to a service for one transport protocol where it is already (or can be subsequently) assigned to a different service for other transport protocols.

The most critical issue in picking a number is selecting the desired range, i.e., System vs. User port numbers. The distinction was intended to indicate a difference in privilege; originally, System port numbers required privileged ('root') access, while User port numbers did not. That distinction has since blurred because some current systems do not limit access control to System port numbers and because some System services have been replicated on User numbers (e.g., IRC). Even so, System port number assignments have continued at an average rate of 3-4 per year over the past 7 years (2007-2013), indicating that the desire to keep this distinction continues.

As a result, the difference between System and User port numbers needs to be treated with caution. Developers are advised to treat services as if they are always run without privilege.

Even when developers seek a System port number assignment, it may be very difficult to obtain. System port number assignment requires IETF Review or IESG Approval and justification that both User and

Dynamic port number ranges are insufficient [RFC6335]. Thus this document recommends both:

>> Developers SHOULD NOT apply for System port number assignments because the increased privilege they are intended to provide is not always enforced.

>> System implementers SHOULD enforce the need for privilege for processes to listen on System port numbers.

At some future date, it might be useful to deprecate the distinction between System and User port numbers altogether. Services typically require elevated ('root') privileges to bind to a System port number, but many such services go to great lengths to immediately drop those privileges just after connection or other association establishment to reduce the impact of an attack using their capabilities. Such services might be more securely operated on User port numbers than on System port numbers. Further, if System port numbers were no longer assigned, as of 2014 it would cost only 180 of the 1024 System values (17%), or 180 of the overall 49152 assigned (System and User) values (<0.04%).

7.4. Support for Security

Just as a service is a way to obtain information or processing from a host over a network, a service can also be the opening through which to compromise that host. Protecting a service involves security, which includes integrity protection, source authentication, privacy, or any combination of these capabilities. Security can be provided in a number of ways, and thus:

>> New services SHOULD support security capabilities, either directly or via a content protection such as TLS [RFC5246] or DTLS [RFC6347] or transport protection such as TCP-AO [RFC5925]. Insecure versions of new or existing secure services SHOULD be avoided because of the new vulnerability they create.

Secure versions of legacy services that are not already security-capable via in-band negotiations can be very useful. However, there is no IETF consensus on when separate ports should be used for secure and insecure variants of the same service [RFC2595] [RFC2817] [RFC6335]. The overall preference is for use of a single port, as noted in Section 6 of this document and Section 7.2 of [RFC6335], but the appropriate approach depends on the specific characteristics of the service. As a result:

>> When requesting both secure and insecure port assignments for the same service, justification is expected for the utility and safety of each port as an independent service (Section 6). Precedent (e.g., citing other protocols that use a separate insecure port) is inadequate justification by itself.

It's also important to recognize that port number assignment is not itself a guarantee that traffic using that number provides the corresponding service, or that a given service is always offered only on its assigned port number. Port numbers are ultimately meaningful only between endpoints and any service can be run on any port. Thus:

>> Security SHOULD NOT rely on assigned port number distinctions alone; every service, whether secure or not, is likely to be attacked.

Applications for a new service that requires both a secure and insecure port may be found, on expert review, to be unacceptable, and may not be approved for allocation. Similarly, an application for a new port to support an insecure variant of an existing secure protocol may be found unacceptable. In both cases, the resulting security of the service in practice will be a significant consideration in the decision as to whether to assign an insecure port.

7.5. Support for Future Versions

Requests for assigned port numbers are expected to support multiple versions on the same assigned port number [RFC6335]. Versions are typically indicated in-band, either at the beginning of a connection or other association, or in each protocol message.

>> Version support SHOULD be included in new services rather than relying on different port number assignments for different versions.

>> Version numbers SHOULD NOT be included in either the service name or service description, to avoid the need to make additional port number assignments for future variants of a service.

Again, the assigned port number space is far too limited to be used as an indicator of protocol version or message type. Although this has happened in the past (e.g., for NFS), it should be avoided in new requests.

7.6. Transport Protocols

IANA assigns port numbers specific to one or more transport protocols, typically UDP [RFC768] and TCP [RFC793], but also SCTP [RFC4960], DCCP [RFC4340], and any other standard transport protocol. Originally, IANA port number assignments were concurrent for both UDP and TCP, and other transports were not indicated. However, to conserve the assigned port number space and to reflect increasing use of other transports, assignments are now specific only to the transport being used.

In general, a service should request assignments for multiple transports using the same service name and description on the same port number only when they all reflect essentially the same service. Good examples of such use are DNS and NFS, where the difference between the UDP and TCP services are specific to supporting each transport. E.g., the UDP variant of a service might add sequence numbers and the TCP variant of the same service might add in-band message delimiters. This document does not describe the appropriate selection of a transport protocol for a service.

>> Service names and descriptions for multiple transport port number assignments SHOULD match only when they describe the same service, excepting only enhancements for each supported transport.

When the services differ, it may be acceptable or preferable to use the same port number, but the service names and descriptions should be different for each transport/service pair, reflecting the differences in the services. E.g., if TCP is used for the basic control protocol and UDP for an alarm protocol, then the services might be "name-ctl" and "name-alarm". A common example is when TCP is used for a service and UDP is used to determine whether that service is active (e.g., via a unicast, broadcast, or multicast test message) [RFC1122]. IANA has, for several years, used the suffix "-disc" in service names to distinguish discovery services, such as are used to identify endpoints capable of a given service:

>> Names of discovery services SHOULD use an identifiable suffix; the suggestion is "-disc".

Some services are used for discovery, either in conjunction with a TCP service or as a stand-alone capability. Such services will be more reliable when using multicast rather than broadcast (over IPv4) because IP routers do not forward "all nodes" broadcasts (all 1's, i.e., 255.255.255.255 for IPv4) and have not been required to support subnet-directed broadcasts since 1999 [RFC1812] [RFC2644].

This issue is relevant only for IPv4 because IPv6 does not support broadcast.

>> UDP over IPv4 multi-host services SHOULD use multicast rather than broadcast.

Designers should be very careful in creating services over transports that do not support congestion control or error recovery, notably UDP. There are several issues that should be considered in such cases, as summarized in Table 1 in [RFC5405]. In addition, the following recommendations apply to service design:

>> Services that use multipoint communication SHOULD be scalable, and SHOULD NOT rely solely on the efficiency of multicast transmission for scalability.

>> Services SHOULD NOT use UDP as a performance enhancement over TCP, e.g., to circumnavigate TCP's congestion control.

7.7. When to Request an Assignment

Assignments are typically requested when a user has enough information to reasonably answer the questions in the IANA application. IANA applications typically take up to a few weeks to process, with some complex cases taking up to a month. The process typically involves a few exchanges between the IANA Ports Expert Review team and the applicant.

An application needs to include a description of the service, as well as to address key questions designed to help IANA determine whether the assignment is justified. The application should be complete and not refer solely to the Internet Draft, RFC, a website, or any other external documentation.

Services that are independently developed can be requested at any time, but are typically best requested in the last stages of design and initial experimentation, before any deployment has occurred that cannot easily be updated.

>> Users MUST NOT deploy implementations that use assigned port numbers prior their assignment by IANA.

>> Users MUST NOT deploy implementations that default to using the experimental System port numbers (1021 and 1022 [RFC4727]) outside a controlled environment where they can be updated with a subsequent assigned port [RFC3692].

Deployments that use unassigned port numbers before assignment complicate IANA management of the port number space. Keep in mind that this recommendation protects existing assignees, users of current services, and applicants for new assignments; it helps ensure that a desired number and service name are available when assigned. The list of currently unassigned numbers is just that - **currently** unassigned. It does not reflect pending applications. Waiting for an official IANA assignment reduces the chance that an assignment request will conflict with another deployed service.

Applications made through Internet Draft / RFC publication (in any stream) typically use a placeholder ("PORTNUM") in the text, and implementations use an experimental port number until a final assignment has been made [RFC6335]. That assignment is initially indicated in the IANA Considerations section of the document, which is tracked by the RFC Editor. When a document has been approved for publication, that request is forwarded to IANA for handling. IANA will make the new assignment accordingly. At that time, IANA may also request that the applicant fill out the application form on their website, e.g., when the RFC does not directly address the information expected as per [RFC6335]. "Early" assignments can be made when justified, e.g., for early interoperability testing, according to existing process [RFC7120] [RFC6335].

>> Users writing specifications SHOULD use symbolic names for port numbers and service names until an IANA assignment has been completed. Implementations SHOULD use experimental port numbers during this time, but those numbers MUST NOT be cited in documentation except as interim.

7.8. Squatting

"Squatting" describes the use of a number from the assignable range in deployed software without IANA assignment for that use, regardless of whether the number has been assigned or remains available for assignment. It is hazardous because IANA cannot track such usage and thus cannot avoid making legitimate assignments that conflict with such unauthorized usage.

Such "squatted" port numbers remain unassigned, and IANA retains the right to assign them when requested by other applicants. Application and service designers are reminded that it is never appropriate to use port numbers that have not been directly assigned [RFC6335]. In particular, any unassigned code from the assigned ranges will be assigned by IANA, and any conflict will be easily resolved as the protocol designer's fault once that happens (because they would not be the assignee). This may reflect in the public's judgment on the

quality of their expertise and cooperation with the Internet community.

Regardless, there are numerous services that have squatted on such numbers that are in widespread use. Designers who are using such port numbers are encouraged to apply for an assignment. Note that even widespread de-facto use may not justify a later IANA assignment of that value, especially if either the value has already been assigned to a legitimate applicant or if the service would not qualify for an assignment of its own accord.

7.9. Other Considerations

As noted earlier, System port numbers should be used sparingly, and it is better to avoid them altogether. This avoids the potentially incorrect assumption that the service on such port numbers run in a privileged mode.

Assigned port numbers are not intended to be changed; this includes the corresponding service name. Once deployed, it can be very difficult to recall every implementation, so the assignment should be retained. However, in cases where the current assignee of a name or number has reasonable knowledge of the impact on such uses, and is willing to accept that impact, the name or number of an assignment can be changed [RFC6335]

Aliases, or multiple service names for the same assigned port number, are no longer considered appropriate [RFC6335].

8. Security Considerations

This document focuses on the issues arising when designing services that require new port assignments. Section 7.4 addresses the security and security-related issues of that interaction.

When designing a secure service, the use of TLS [RFC5246], DTLS [RFC6347], or TCP-AO [RFC5925] mechanisms that protect transport protocols or their contents is encouraged. It may not be possible to use IPsec [RFC4301] in similar ways because of the different relationship between IPsec and port numbers and because applications may not be aware of IPsec protections.

This document reminds application and service designers that port numbers do not protect against denial of service attack or guarantee that traffic should be trusted. Using assigned numbers for port filtering isn't a substitute for authentication, encryption, and integrity protection. The port number alone should not be used to

avoid denial of service attacks or to manage firewall traffic because the use of port numbers is not regulated or validated.

The use of assigned port numbers is the antithesis of privacy because they are intended to explicitly indicate the desired application or service. Strictly, port numbers are meaningful only at the endpoints, so any interpretation elsewhere in the network can be arbitrarily incorrect. However, those numbers can also expose information about available services on a given host. This information can be used by intermediate devices to monitor and intercept traffic as well as to potentially identify key endpoint software properties ("fingerprinting"), which can be used to direct other attacks.

9. IANA Considerations

The entirety of this document focuses on suggestions that help ensure the conservation of port numbers and provide useful hints for issuing informative requests thereof.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2780] Bradner, S., and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.
- [RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3962, Jan. 2004.
- [RFC4727] Fenner, B., "Experimental Values in IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, November 2006.
- [RFC5246] Dierks, T., and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5405] Eggert, L., and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, Nov. 2008.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

- [RFC6335] Cotton, M., L. Eggert, J. Touch, M. Westerlund, and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.
- [RFC6347] Rescorla, E., and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

10.2. Informative References

- [IEN112] Postel, J., "Transmission Control Protocol", IEN 112, August 1979.
- [RFC33] Crocker, S., "New Host-Host Protocol", RFC 33 February 1970.
- [RFC37] Crocker, S., "Network Meeting Epilogue", RFC 37, March 1970.
- [RFC38] Wolfe, S., "Comments on Network Protocol from NWG/RFC #36", RFC 38, March 1970.
- [RFC48] Postel, J., and S. Crocker, "Possible protocol plateau", RFC 48, April 1970.
- [RFC61] Walden, D., "Note on Interprocess Communication in a Resource Sharing Computer Network", RFC 61, July 1970.
- [RFC76] Bouknight, J., J. Madden, and G. Grossman, "Connection by name: User oriented protocol", RFC 76, October 1970.
- [RFC333] Bressler, R., D. Murphy, and D. Walden. "Proposed experiment with a Message Switching Protocol", RFC 333, May 1972.
- [RFC739] Postel, J., "Assigned numbers", RFC 739, November 1977.
- [RFC758] Postel, J., "Assigned numbers", RFC 758, August 1979.
- [RFC768] Postel, J., "User Datagram Protocol", RFC 768, August 1980.
- [RFC793] Postel, J., "Transmission Control Protocol" RFC 793, September 1981
- [RFC820] Postel, J., "Assigned numbers", RFC 820, August 1982.

- [RFC900] Reynolds, J., and J. Postel, "Assigned numbers", RFC 900, June 1984.
- [RFC959] Postel, J., and J. Reynolds, "FILE TRANSFER PROTOCOL (FTP)", RFC 959, October 1985.
- [RFC1122] Braden, B. (Ed.), "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989.
- [RFC1340] Reynolds, J., and J. Postel, "Assigned numbers", RFC 1340, July 1992.
- [RFC1700] Reynolds, J., and J. Postel, "Assigned numbers", RFC 1700, October 1994.
- [RFC1812] Baker, F. (Ed.), "Requirements for IP Version 4 Routers", RFC 1812, June 1995.
- [RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2", RFC 1833, August 1995.
- [RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999.
- [RFC2644] Senie, D., "Changing the Default for Directed Broadcasts in Routers", RFC 2644, August 1999.
- [RFC2817] Khare, R., and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.
- [RFC3232] Reynolds, J. (Ed.), "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, January 2002.
- [RFC3261] Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC4301] Kent, S., and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4340] Kohler, E., M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC4960] Stewart, R. (Ed.), "Stream Control Transmission Protocol", RFC 4960, September 2007.

- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5389] Rosenberg, J., R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT", RFC 5389, October 2008.
- [RFC5766] Mahy, R., P. Matthews, and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.
- [RFC6762] Cheshire, S., and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.
- [RFC6763] Cheshire, S., and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.
- [RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, January 2014.
- [RFC7230] Fielding, R., (Ed.), and J. Reshke, (Ed.), "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

11. Acknowledgments

This work benefitted from the feedback from David Black, Lars Eggert, Gorry Fairhurst, and Eliot Lear, as well as discussions of the IETF TSVWG WG.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292-6695
U.S.A.

Phone: +1 (310) 448-9151
EMail: touch@isi.edu

Network WG
Internet-Draft
Expires: January 4, 2015
Intended Status: Standards Track
Updates: RFC 2872 (if accepted)

James Polk
Subha Dhesikan
Cisco Systems
July 4, 2014

Resource Reservation Protocol (RSVP) Application-ID
Profiles for Voice and Video Streams
draft-ietf-tsvwg-rsvp-app-id-vv-profiles-02

Abstract

RFC 2872 defines an Resource Reservation Protocol (RSVP) object for application identifiers. This document uses that App-ID and gives implementers specific guidelines for differing voice and video stream identifications to nodes along a reservation path, creating specific profiles for voice and video session identification.

Status of this Memo

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 July 4, 2014.

Copyright Notice

Copyright (c) 2014 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.	RSVP Application-ID Template	3
3.	The Voice and Video Application-ID Profiles	4
3.1	The Broadcast video Profile	4
3.2	The Real-time Interactive Profile	5
3.3	The Multimedia Conferencing Profile	5
3.4	The Multimedia Streaming Profile	6
3.5	The Conversational Profile	6
4.	Security considerations	7
5.	IANA considerations	7
5.1	Application Profiles	7
5.1.1	Broadcast Profiles IANA Registry	8
5.1.2	Realtime-Interactive Profiles IANA Registry	8
5.1.3	Multimedia-Conferencing Profiles IANA Registry	9
5.1.4	Multimedia-Streaming Profiles IANA Registry	10
5.1.5	Conversational Profiles IANA Registry	10
6.	Acknowledgments	12
7.	References	12
7.1.	Normative References	12
7.2.	Informative References	13
	Authors' Addresses	13
	Appendix	14

1. Introduction

RFC 2872 [RFC2872] describes the usage of policy elements for providing application information in Resource Reservation Protocol (RSVP) signaling [RFC2205]. The intention of providing this information is to enable application-based policy control. However, RFC 2872 does not enumerate any application profiles. The absence of explicit, uniform profiles leads to incompatible handling of these values and misapplied policies. An application profile used by a sender might not be understood by the intermediaries or receiver in a different domain. Therefore, there is a need to enumerate application profiles that are universally understood and applied for correct policy control.

Call control between endpoints has the ability to bind or associate many attributes to a reservation. One new attribute is currently being defined so as to establish the type of traffic contained in that reservation. This is accomplished via assigning a traffic label to the call (or session or flow) [ID-TRAF-CLASS].

This document takes the application traffic classes from [ID-TRAF-CLASS] and places those strings in the APP-ID object defined in RFC 2872. Thus, the intermediary devices (e.g., routers) processing the RSVP message can learn the identified profile within the Application-ID policy element for a particular reservation, and possibly be configured with the profile(s) to understand them

correctly, thus performing the correct admission control.

Another goal of this document is to the ability to signal an application profile which can then be translated into a DSCP value as per the choice of each domain. While the DCLASS object [RFC2996] allows the transfer of DSCP value in an RSVP message, that RFC does not allow the flexibility of having different domains choosing the DSCP value for the traffic classes that they maintain.

How these labels indicate the appropriate Differentiated Services Codepoint (DSCP) is out of scope for this document.

This document will break out each application type and propose how the values in application-id template should be populated for uniformity and interoperability.

1.1 Terminology

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

2. RSVP Application ID Template

The template from RFC 2872 is as follows:

0	1	2	3
PE Length (8)	P-type = AUTH_APP		
Attribute Length	A-type = POLICY_LOCATOR	Sub-type = ASCII_DN	
Application name as ASCII string (e.g. SAP.EXE)			

In line with how this policy element is constructed in RFC 2872, the A-type will remain "POLICY_LOCATOR".

The P-type field is first created in [RFC2752]. This document uses the existing P-type "AUTH_APP" for application traffic class.

The first Sub-type will be mandatory for every profile within this document, and will be "ASCII_DN". No other Sub-types are defined by any profile within this document, but MAY be included by individual implementations - and MUST be ignored if not understood by receiving implementations along the reservation path.

RFC 2872 states the #1 sub-element from RFC 2872 as the "identifier that uniquely identifies the application vendor", which is optional to include. This document modifies this vendor limitation so that the identifier need only be unique - and not limited to an application vendor (identifier). For example, this specification now allows an RFC that defines an industry recognizable term or string to be a valid identifier. For example, a term or string taken from another IETF document, such as "conversational" or "avconf" from [ID-TRAF-CLASS]. This sub-element is still optional to include.

The following subsections will define the values within the above template into specific profiles for voice and video identification.

3. The Voice and Video Application-ID Profiles

This section contains the elements of the Application ID policy object which is used to signal the application classes defined in [ID-TRAF-CLASS].

3.1 The Broadcast Profiles

Broadcast profiles are for minimally buffered one-way streaming flows, such as video surveillance, or Internet based concerts or non-VOD TV broadcasts such as live sporting events.

This document creates Broadcast profiles for

- Broadcast IPTV for audio and video
- Broadcast Live-events for audio and video
- Broadcast Surveillance for audio and video

Here is an example profile for identifying Broadcast Video-Surveillance

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,  
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
APP=broadcast.video.surveillance, VER="
```

[Editor's Note: "rfcXXXX" will be replaced with the RFC number assigned to the [ID-TRAF-CLASS] reference. This 'note' should be removed during the RFC-Editor review process.]

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value at this time.

3.2 The Realtime Interactive Profiles

Realtime Interactive profiles are for on-line gaming, and both remote and virtual avconf applications, in which the timing is particularly important towards the feedback to uses of these applications. This traffic type will generally not be UDP based, with minimal tolerance to RTT delays.

This document creates Realtime Interactive profiles for

- Realtime-Interactive Gaming
- Realtime-Interactive Remote-Desktop
- Realtime-Interactive Virtualized-Desktop

Here is the profile for identifying Realtime-Interactive Gaming

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,  
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
APP=realtime-interactive.gaming, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.3 The Multimedia Conferencing Profiles

There will be Multimedia Conferencing profiles for presentation data, application sharing and whiteboarding, where these applications will most often be associated with a larger Conversational (audio and/or audio/video) conference. Timing is important, but some minimal delays are acceptable, unlike the case for Realtime-Interactive traffic.

This document creates Multimedia-Conferencing profiles for

- Multimedia-Conferencing presentation-data
- Multimedia-Conferencing presentation-video
- Multimedia-Conferencing presentation-audio
- Multimedia-Conferencing application-sharing
- Multimedia-Conferencing whiteboarding

Here is the profile for identifying Multimedia-Conferencing Application-sharing

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,  
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
APP=multimedia-conferencing.application-sharing, VER="
```

Where the Globally Unique Identifier (GUID) indicates the RFC reference that created this well-known string [ID-TRAF-CLASS], the

APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.4 The Multimedia Streaming Profiles

Multimedia Streaming profiles are for more significantly buffered one-way streaming flows than Broadcast profiles. These include...

This document creates Multimedia Streaming profiles for

- Multimedia-Streaming multiplex
- Multimedia-Streaming webcast

Here is the profile for identifying Multimedia Streaming webcast

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,  
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
APP=multimedia-streaming.webcast, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.5 The Conversational Profiles

Conversational category is for realtime bidirectional communications, such as voice or video, and is the most numerous due to the choices of application with or without adjectives. The number of profiles is then doubled because there needs to be one for unadmitted and one for admitted. The IANA section lists all that are currently proposed for registration at this time, therefore there will not be an exhaustive list provided in this section.

This document creates Conversational profiles for

- Conversational Audio
- Conversational Audio Admitted
- Conversational Video
- Conversational Video Admitted
- Conversational Audio Avconf
- Conversational Audio Avconf Admitted
- Conversational Video Avconf
- Conversational Video Avconf Admitted
- Conversational Audio Immersive
- Conversational Audio Immersive Admitted
- Conversational Video Immersive
- Conversational Video Immersive Admitted

Here is an example profile for identifying Conversational Audio:

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,  
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
APP=conversational.audio, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

4. Security considerations

The security considerations section within RFC 2872 sufficiently covers this document, with one possible exception - someone using the wrong template values (e.g., claiming a reservation is Multimedia Streaming when it is in fact Real-time Interactive). Given that each traffic flow is within separate reservations, and RSVP does not have the ability to police the type of traffic within any reservation, solving for this appears to be administratively handled at best. This is not meant to be a 'punt', but there really is nothing this template creates that is going to make things any harder for anyone (that we know of now).

5. IANA considerations

5.1 Application Profiles

This document requests IANA create a new registry for the application identification classes similar to the following table within the Resource Reservation Protocol (RSVP) Parameters registry:

```
Registry Name: RSVP APP-ID Profiles  
Reference: [this document]  
Registration procedures: Standards Track document [RFC5226]
```

```
[Editor's Note: "rfcXXXX" will be replaced with the RFC number  
assigned to the [ID-TRAF-CLASS] reference. This  
'note' should be removed during the RFC-Editor  
review process.]
```

5.1.1 Broadcast Profiles IANA Registry

Broadcast Audio IPTV Profile

```
P-type = AUTH_APP  
A-type = POLICY_LOCATOR  
Sub-type = ASCII_DN  
Conformant policy locator =  
    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
    APP=broadcast.audio.iptv, VER="  
Reference: [this document]
```

Broadcast Video IPTV Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.video.iptv, VER="

Reference: [this document]

Broadcast Audio Live-events Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.audio.live-events, VER="

Reference: [this document]

Broadcast Video Live-events Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.video.live-events, VER="

Reference: [this document]

Broadcast Audio-Surveillance Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.audio.surveillance, VER="

Reference: [this document]

Broadcast Video-Surveillance Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.video.surveillance, VER="

Reference: [this document]

5.1.2 Realtime-Interactive Profiles IANA Registry

Realtime-Interactive Gaming Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP= realtime-interactive.gaming, VER="

Reference: [this document]

Real-time Interactive Remote-Desktop Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=realtime-interactive.remote-desktop, VER="

Reference: [this document]

Real-time Interactive Virtualized-Desktop Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=realtime-interactive.
remote-desktop.virtual, VER="

Reference: [this document]

Real-time Interactive Telemetry Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=realtime-interactive.telemetry, VER="

Reference: [this document]

5.1.3 Multimedia-Conferencing Profiles IANA Registry

Multimedia-Conferencing Presentation-Data Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP= multimedia-conferencing.presentation-data,
VER="

Reference: [this document]

Multimedia-Conferencing Presentation-Video Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,

APP= multimedia-conferencing.presentation-video,
VER="

Reference: [this document]

Multimedia-Conferencing Presentation-Audio Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP= multimedia-conferencing.presentation-audio,
VER="

Reference: [this document]

Multimedia-Conferencing Application-Sharing Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP= multimedia-conferencing.application-sharing,
VER="

Reference: [this document]

Multimedia-Conferencing Whiteboarding Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP= multimedia-conferencing.whiteboarding, VER="

Reference: [this document]

5.1.4 Multimedia-Streaming Profiles IANA Registry

Multimedia-Streaming Multiplex Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=multimedia-streaming.multiplex, VER="

Reference: [this document]

Multimedia-Streaming Webcast Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=multimedia-streaming.webcast, VER="

Reference: [this document]

5.1.5 Conversational Profiles IANA Registry

Conversational Audio Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.audio, VER="

Reference: [this document]

Conversational Audio Admitted Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.audio.aq:admitted, VER="

Reference: [this document]

Conversational Video Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.video, VER="

Reference: [this document]

Conversational Video Admitted Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.video.aq:admitted, VER="

Reference: [this document]

Conversational Audio Avconf Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.audio.avconf, VER="

Reference: [this document]

Conversational Audio Avconf Admitted Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
 APP=conversational.audio.avconf.aq:admitted,
 VER="

Reference: [this document]

Conversational Video Avconf Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
 APP=conversational.video.avconf, VER="

Reference: [this document]

Conversational Video Avconf Admitted Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
 APP=conversational.video.avconf.aq:admitted,
 VER="

Reference: [this document]

Conversational Audio Immersive Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
 APP=conversational.audio.immersive, VER="

Reference: [this document]

Conversational Audio Immersive Admitted Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
 APP=conversational.audio.immersive.aq:admitted,
 VER="

Reference: [this document]

Conversational Video Immersive Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,

APP=conversational.video.immersive, VER="

Reference: [this document]

Conversational Video Immersive Admitted Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.video.immersive.aq:admitted,
VER="

Reference: [this document]

6. Acknowledgments

To Francois Le Faucheur, Paul Jones, Ken Carlberg, Georgios Karagiannis and Glen Lavers for their helpful comments, document reviews and encouragement.

7. References

7.1. Normative References

- [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997
- [RFC2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997
- [RFC2474] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers ", RFC 2474, December 1998
- [RFC2750] S. Herzog, "RSVP Extensions for Policy Control", RFC 2750, January 2000
- [RFC2872] Y. Bernet, R. Pabbati, "Application and Sub Application Identity Policy Element for Use with RSVP", RFC 2872, June 2000
- [RFC2996] Y. Bernet, "Format of the RSVP DCLASS Object", RFC 2996, November 2000
- [RFC3182] S. Yadav, R. Yavatkar, R. Pabbati, P. Ford, T. Moore, S. Herzog, R. Hess, "Identity Representation for RSVP", RFC 3182, October 2001
- [RFC5226] T. Narten, H. Alvestrand, "Guidelines for Writing an IANA

Considerations Section in RFCs", RFC 5226, May 2008

[ID-TRAF-CLASS] J. Polk, S. Dhesikan, P. Jones, "The Session Description Protocol (SDP) 'trafficclass' Attribute", work in progress, Feb 2013

7.2. Informative References

[RFC4594] J. Babiarez, K. Chan, F Baker, "Configuration Guidelines for Diffserv Service Classes", RFC 4594, August 2006

Authors' Addresses

James Polk
3913 Treemont Circle
Colleyville, Texas, USA
+1.817.271.3552

mailto: jmpolk@cisco.com

Subha Dhesikan
170 W Tasman St
San Jose, CA, USA
+1.408-902-3351

mailto: sdhesika@cisco.com

Appendix - Changes to ID

[Editor's Note: this appendix should be removed in the RFC-Editor's process.]

A.1 - Changes from WG version -00 to WG version -01

The following changes were made in this version:

- corrected nits
- globally replaced GUID link from the MMUSIC Trafficclass ID to the future RFC of that document.
- added profiles for presentation-video and presentation-audio

A.2 - Changes from Individual -04 to WG version -00

The following changes were made in this version:

- changed P-Type from APP_TC back to AUTH_APP, which is already defined.
- fixed nits and inconsistencies

A.3 - Changes from Individual -03 to -04

The following changes were made in this version:

- clarified security considerations section to mean RSVP cannot police the type of traffic within a reservation to know if a traffic flow should be using a different profile, as defined in this document.
- changed existing informative language regarding "... other Sub-types ..." from 'can' to normative 'MAY'.
- editorial changes to clear up minor mistakes

A.4 - Changes from Individual -02 to -03

The following changes were made in this version:

- Added [ID-TRAF-CLASS] as a reference
- Changed to a new format of the profile string.
- Added many new profiles based on the new format into each parent category of Section 3.
- changed the GUID to refer to draft-ietf-mmusic-traffic-class-for-sdp-03.txt
- changed 'desktop' adjective to 'avconf' to keep in alignment with [ID-TRAF-CLASS]
- Have a complete IANA Registry proposal for each application-ID discussed in this draft.
- General text clean-up of the draft.

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 20, 2017

P. Jones
S. Dhesikan
C. Jennings
Cisco Systems
D. Druta
AT&T
August 19, 2016

DSCP Packet Markings for WebRTC QoS
draft-ietf-tsvwg-rtcweb-qos-18

Abstract

Many networks, such as service provider and enterprise networks, can provide different forwarding treatments for individual packets based on Differentiated Services Code Point (DSCP) values on a per-hop basis. This document provides the recommended DSCP values for web browsers to use for various classes of WebRTC traffic.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 20, 2017.

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. Terminology	3
3. Relation to Other Specifications	3
4. Inputs	4
5. DSCP Mappings	5
6. Security Considerations	8
7. IANA Considerations	8
8. Downward References	9
9. Acknowledgements	9
10. Dedication	9
11. Document History	9
12. References	9
12.1. Normative References	9
12.2. Informative References	10
Authors' Addresses	11

1. Introduction

Differentiated Services Code Point (DSCP) [RFC2474] packet marking can help provide QoS in some environments. This specification provides default packet marking for browsers that support WebRTC applications, but does not change any advice or requirements in other IETF RFCs. The contents of this specification are intended to be a simple set of implementation recommendations based on the previous RFCs.

Networks where these DSCP markings are beneficial (likely to improve QoS for WebRTC traffic) include:

1. Private, wide-area networks. Network administrators have control over remarking packets and treatment of packets.
2. Residential Networks. If the congested link is the broadband uplink in a cable or DSL scenario, often residential routers/NAT support preferential treatment based on DSCP.
3. Wireless Networks. If the congested link is a local wireless network, marking may help.

There are cases where these DSCP markings do not help, but, aside from possible priority inversion for "less than best effort traffic"

(see Section 5), they seldom make things worse if packets are marked appropriately.

DSCP values are in principle site specific, with each site selecting its own code points for controlling per-hop-behavior to influence the QoS for transport-layer flows. However in the WebRTC use cases, the browsers need to set them to something when there is no site specific information. This document describes a subset of DSCP code point values drawn from existing RFCs and common usage for use with WebRTC applications. These code points are intended to be the default values used by a WebRTC application. While other values could be used, using a non-default value may result in unexpected per-hop behavior. It is RECOMMENDED that WebRTC applications use non-default values only in private networks that are configured to use different values.

This specification defines inputs that are provided by the WebRTC application hosted in the browser that aid the browser in determining how to set the various packet markings. The specification also defines the mapping from abstract QoS policies (flow type, priority level) to those packet markings.

2. Terminology

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

The terms "browser" and "non-browser" are defined in [RFC7742] and carry the same meaning in this document.

3. Relation to Other Specifications

This document is a complement to [RFC7657], which describes the interaction between DSCP and real-time communications. That RFC covers the implications of using various DSCP values, particularly focusing on Real-time Transport Protocol (RTP) [RFC3550] streams that are multiplexed onto a single transport-layer flow.

There are a number of guidelines specified in [RFC7657] that apply to marking traffic sent by WebRTC applications, as it is common for multiple RTP streams to be multiplexed on the same transport-layer flow. Generally, the RTP streams would be marked with a value as appropriate from Table 1. A WebRTC application might also multiplex data channel [I-D.ietf-rtcweb-data-channel] traffic over the same 5-tuple as RTP streams, which would also be marked as per that table. The guidance in [RFC7657] says that all data channel traffic would be marked with a single value that is typically different than the

value(s) used for RTP streams multiplexed with the data channel traffic over the same 5-tuple, assuming RTP streams are marked with a value other than default forwarding (DF). This is expanded upon further in the next section.

This specification does not change or override the advice in any other IETF RFCs about setting packet markings. Rather, it simply selects a subset of DSCP values that is relevant in the WebRTC context.

The DSCP value set by the endpoint is not trusted by the network. In addition, the DSCP value may be remarked at any place in the network for a variety of reasons to any other DSCP value, including default forwarding (DF) value to provide basic best effort service. Even so, there is benefit in marking traffic even if it only benefits the first few hops. The implications are discussed in Section 3.2 of [RFC7657]. Further, a mitigation for such action is through an authorization mechanism. Such an authorization mechanism is outside the scope of this document.

4. Inputs

WebRTC applications send and receive two types of flows of significance to this document:

- o media flows which are RTP streams [I-D.ietf-rtcweb-rtp-usage]
- o data flows which are data channels [I-D.ietf-rtcweb-data-channel]

Each of the RTP streams and distinct data channels consists of all of the packets associated with an independent media entity, so an RTP stream or distinct data channel is not always equivalent to a transport-layer flow defined by a 5-tuple (source address, destination address, source port, destination port, and protocol). There may be multiple RTP streams and data channels multiplexed over the same 5-tuple, with each having a different level of importance to the application and, therefore, potentially marked using different DSCP values than another RTP stream or data channel within the same transport-layer flow. (Note that there are restrictions with respect to marking different data channels carried within the same SCTP association as outlined in Section 5.)

The following are the inputs provided by the WebRTC application to the browser:

- o Flow Type: The application provides this input because it knows if the flow is audio, interactive video [RFC4594] [G.1010] with or without audio, or data.

- o Application Priority: Another input is the relative importance of an RTP stream or data channel. Many applications have multiple flows of the same Flow Type and often some flows are more important than others. For example, in a video conference where there are usually audio and video flows, the audio flow may be more important than the video flow. JavaScript applications can tell the browser whether a particular flow is high, medium, low or very low importance to the application.

[I-D.ietf-rtcweb-transports] defines in more detail what an individual flow is within the WebRTC context and priorities for media and data flows.

Currently in WebRTC, media sent over RTP is assumed to be interactive [I-D.ietf-rtcweb-transports] and browser APIs do not exist to allow an application to differentiate between interactive and non-interactive video.

5. DSCP Mappings

The DSCP values for each flow type of interest to WebRTC based on application priority are shown in Table 1. These values are based on the framework and recommended values in [RFC4594]. A web browser SHOULD use these values to mark the appropriate media packets. More information on EF can be found in [RFC3246]. More information on AF can be found in [RFC2597]. DF is default forwarding which provides the basic best effort service [RFC2474].

WebRTC use of multiple DSCP values may encounter network blocking of packets with certain DSCP values. See section 4.2 of [I-D.ietf-rtcweb-transports] for further discussion, including how WebRTC implementations establish and maintain connectivity when such blocking is encountered.

Flow Type	Very Low	Low	Medium	High
Audio	CS1 (8)	DF (0)	EF (46)	EF (46)
Interactive Video with or without Audio	CS1 (8)	DF (0)	AF42, AF43 (36, 38)	AF41, AF42 (34, 36)
Non-Interactive Video with or without Audio	CS1 (8)	DF (0)	AF32, AF33 (28, 30)	AF31, AF32 (26, 28)
Data	CS1 (8)	DF (0)	AF11	AF21

Table 1: Recommended DSCP Values for WebRTC Applications

The application priority, indicated by the columns "very low", "low", "Medium", and "high", signifies the relative importance of the flow within the application. It is an input that the browser receives to assist in selecting the DSCP value and adjusting the network transport behavior.

The above table assumes that packets marked with CS1 are treated as "less than best effort", such as the LE behavior described in [RFC3662]. However, the treatment of CS1 is implementation dependent. If an implementation treats CS1 as other than "less than best effort", then the actual priority (or, more precisely, the per-hop-behavior) of the packets may be changed from what is intended. It is common for CS1 to be treated the same as DF, so applications and browsers using CS1 cannot assume that CS1 will be treated differently than DF [RFC7657]. However, it is also possible per [RFC2474] for CS1 traffic to be given better treatment than DF, thus caution should be exercised when electing to use CS1. This is one of the cases where marking packets using these recommendations can make things worse.

Implementers should also note that excess EF traffic is dropped. This could mean that a packet marked as EF may not get through, although the same packet marked with a different DSCP value would have gotten through. This is not a flaw, but how excess EF traffic is intended to be treated.

The browser SHOULD first select the flow type of the flow. Within the flow type, the relative importance of the flow SHOULD be used to select the appropriate DSCP value.

Currently, all WebRTC video is assumed to be interactive [I-D.ietf-rtcweb-transports], for which the Interactive Video DSCP values in Table 1 SHOULD be used. Browsers MUST NOT use the AF3x DSCP values (for Non-Interactive Video in Table 1) for WebRTC applications. Non-browser implementations of WebRTC MAY use the AF3x DSCP values for video that is known not to be interactive, e.g., all video in a WebRTC video playback application that is not implemented in a browser.

The combination of flow type and application priority provides specificity and helps in selecting the right DSCP value for the flow. All packets within a flow SHOULD have the same application priority. In some cases, the selected application priority cell may have multiple DSCP values, such as AF41 and AF42. These offer different drop precedences. The different drop precedence values provides additional granularity in classifying packets within a flow. For example, in a video conference the video flow may have medium application priority, thus either AF42 or AF43 may be selected. More important video packets (e.g., a video picture or frame encoded without any dependency on any prior pictures or frames) might be marked with AF42 and less important packets (e.g., a video picture or frame encoded based on the content of one or more prior pictures or frames) might be marked with AF43 (e.g., receipt of the more important packets enables a video renderer to continue after one or more packets are lost).

It is worth noting that the application priority is utilized by the coupled congestion control mechanism for media flows per [I-D.ietf-rmcat-coupled-cc] and the SCTP scheduler for data channel traffic per [I-D.ietf-rtcweb-data-channel].

For reasons discussed in Section 6 of [RFC7657], if multiple flows are multiplexed using a reliable transport (e.g., TCP) then all of the packets for all flows multiplexed over that transport-layer flow MUST be marked using the same DSCP value. Likewise, all WebRTC data channel packets transmitted over an SCTP association MUST be marked using the same DSCP value, regardless of how many data channels (streams) exist or what kind of traffic is carried over the various SCTP streams. In the event that the browser wishes to change the DSCP value in use for an SCTP association, it MUST reset the SCTP congestion controller after changing values. Frequent changes in the DSCP value used for an SCTP association are discouraged, though, as this would defeat any attempts at effectively managing congestion. It should also be noted that any change in DSCP value that results in a reset of the congestion controller puts the SCTP association back into slow start, which may have undesirable effects on application performance.

For the data channel traffic multiplexed over an SCTP association, it is RECOMMENDED that the DSCP value selected be the one associated with the highest priority requested for all data channels multiplexed over the SCTP association. Likewise, when multiplexing multiple flows over a TCP connection, the DSCP value selected should be the one associated with the highest priority requested for all multiplexed flows.

If a packet enters a network that has no support for a flow type-application priority combination specified in Table 1, then the network node at the edge will remark the DSCP value based on policies. This could result in the flow not getting the network treatment it expects based on the original DSCP value in the packet. Subsequently, if the packet enters a network that supports a larger number of these combinations, there may not be sufficient information in the packet to restore the original markings. Mechanisms for restoring such original DSCP is outside the scope of this document.

In summary, DSCP marking provides neither guarantees nor promised levels of service. However, DSCP marking is expected to provide a statistical improvement in real-time service as a whole. The service provided to a packet is dependent upon the network design along the path, as well as the network conditions at every hop.

6. Security Considerations

Since the JavaScript application specifies the flow type and application priority that determine the media flow DSCP values used by the browser, the browser could consider application use of a large number of higher priority flows to be suspicious. If the server hosting the JavaScript application is compromised, many browsers within the network might simultaneously transmit flows with the same DSCP marking. The DiffServ architecture requires ingress traffic conditioning for reasons that include protecting the network from this sort of attack.

Otherwise, this specification does not add any additional security implications beyond those addressed in the following DSCP-related specifications. For security implications on use of DSCP, please refer to Section 7 of [RFC7657] and Section 6 of [RFC4594]. Please also see [I-D.ietf-rtcweb-security] as an additional reference.

7. IANA Considerations

This specification does not require any actions from IANA.

8. Downward References

This specification contains a downwards reference to [RFC4594] and [RFC7657]. However, the parts of the former RFC used by this specification are sufficiently stable for this downward reference. The guidance in the latter RFC is necessary to understand the Diffserv technology used in this document and the motivation for the recommended DSCP values and procedures.

9. Acknowledgements

Thanks to David Black, Magnus Westerlund, Paolo Severini, Jim Hasselbrook, Joe Marcus, Erik Nordmark, Michael Tuexen, and Brian Carpenter for their invaluable input.

10. Dedication

This document is dedicated to the memory of James Polk, a long-time friend and colleague. James made important contributions to this specification, including serving initially as one of the primary authors. The IETF global community mourns his loss and he will be missed dearly.

11. Document History

Note to RFC Editor: Please remove this section.

This document was originally an individual submission in RTCWeb WG. The RTCWeb working group selected it to become a WG document. Later the transport ADs requested that this be moved to the TSVWG WG as that seemed to be a better match.

12. References

12.1. Normative References

[I-D.ietf-rtcweb-data-channel]

Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ietf-rtcweb-rtp-usage]

Perkins, D., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-26 (work in progress), March 2016.

- [I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", draft-ietf-rtcweb-security-08 (work in progress), February 2015.
- [I-D.ietf-rtcweb-transports]
Alvestrand, H., "Transports for WebRTC", draft-ietf-rtcweb-transports-15 (work in progress), August 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997,
<<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006,
<<http://www.rfc-editor.org/info/rfc4594>>.
- [RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015,
<<http://www.rfc-editor.org/info/rfc7657>>.
- [RFC7742] Roach, A., "WebRTC Video Processing and Codec Requirements", RFC 7742, DOI 10.17487/RFC7742, March 2016,
<<http://www.rfc-editor.org/info/rfc7742>>.

12.2. Informative References

- [G.1010] International Telecommunications Union, "End-user multimedia QoS categories", Recommendation ITU-T G.1010, November 2001.
- [I-D.ietf-rmcat-coupled-cc]
Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-03 (work in progress), July 2016.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998,
<<http://www.rfc-editor.org/info/rfc2474>>.
- [RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, DOI 10.17487/RFC2597, June 1999,
<<http://www.rfc-editor.org/info/rfc2597>>.

- [RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, DOI 10.17487/RFC3246, March 2002, <<http://www.rfc-editor.org/info/rfc3246>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<http://www.rfc-editor.org/info/rfc3550>>.
- [RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services", RFC 3662, DOI 10.17487/RFC3662, December 2003, <<http://www.rfc-editor.org/info/rfc3662>>.

Authors' Addresses

Paul E. Jones
Cisco Systems

Email: paulej@packetizer.com

Subha Dhesikan
Cisco Systems

Email: sdhesika@cisco.com

Cullen Jennings
Cisco Systems

Email: fluffy@cisco.com

Dan Druta
AT&T

Email: dd5826@att.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 28, 2015

M. Tuexen
Muenster Univ. of Appl. Sciences
R. Stewart
Netflix, Inc.
R. Jesup
WorldGate Communications
S. Loreto
Ericsson
January 24, 2015

DTLS Encapsulation of SCTP Packets
draft-ietf-tsvwg-sctp-dtls-encaps-09.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a transport protocol originally defined to run on top of the network protocols IPv4 or IPv6. This document specifies how SCTP can be used on top of the Datagram Transport Layer Security (DTLS) protocol. Using the encapsulation method described in this document, SCTP is unaware of the protocols being used below DTLS; hence explicit IP addresses cannot be used in the SCTP control chunks. As a consequence, the SCTP associations carried over DTLS can only be single homed.

Status of This Memo

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 July 28, 2015.

Copyright Notice

Copyright (c) 2015 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. Overview	2
2. Conventions	3
3. Encapsulation and Decapsulation Procedure	3
4. General Considerations	3
5. DTLS Considerations	4
6. SCTP Considerations	5
7. IANA Considerations	6
8. Security Considerations	6
9. Acknowledgments	7
10. References	7
Appendix A. NOTE to the RFC-Editor	9
Authors' Addresses	9

1. Overview

The Stream Control Transmission Protocol (SCTP) as defined in [RFC4960] is a transport protocol running on top of the network protocols IPv4 [RFC0791] or IPv6 [RFC2460]. This document specifies how SCTP is used on top of the Datagram Transport Layer Security (DTLS) protocol. DTLS 1.0 is defined in [RFC4347] and the latest version when this RFC was published, DTLS 1.2, is defined in [RFC6347]. This encapsulation is used for example within the WebRTC protocol suite (see [I-D.ietf-rtcweb-overview] for an overview) for transporting non-SRTP data between browsers. The architecture of this stack is described in [I-D.ietf-rtcweb-data-channel].

[NOTE to RFC-Editor:

Please ensure that the authors double check the above statement about DTLS 1.2 during AUTH48 and then remove this note before publication.

]

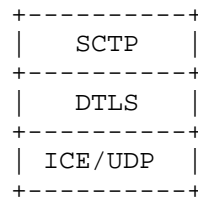


Figure 1: Basic stack diagram

This encapsulation of SCTP over DTLS over UDP or ICE/UDP (see [RFC5245]) can provide a NAT traversal solution in addition to confidentiality, source authentication, and integrity protected transfers. Please note that using ICE does not necessarily imply that a different packet format is used on the wire.

Please note that the procedures defined in [RFC6951] for dealing with the UDP port numbers do not apply here. When using the encapsulation defined in this document, SCTP is unaware about the protocols used below DTLS.

2. Conventions

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

3. Encapsulation and Decapsulation Procedure

When an SCTP packet is provided to the DTLS layer, the complete SCTP packet, consisting of the SCTP common header and a number of SCTP chunks, is handled as the payload of the application layer protocol of DTLS. When the DTLS layer has processed a DTLS record containing a message of the application layer protocol, the payload is passed to the SCTP layer. The SCTP layer expects an SCTP common header followed by a number of SCTP chunks.

4. General Considerations

An implementation of SCTP over DTLS MUST implement and use a path maximum transmission unit (MTU) discovery method that functions without ICMP to provide SCTP/DTLS with an MTU estimate. An implementation of "Packetization Layer Path MTU Discovery" [RFC4821] either in SCTP or DTLS is RECOMMENDED.

The path MTU discovery is performed by SCTP when SCTP over DTLS is used for data channels (see Section 5 of [I-D.ietf-rtcweb-data-channel]).

5. DTLS Considerations

The DTLS implementation MUST support DTLS 1.0 [RFC4347] and SHOULD support the most recently published version of DTLS, which was DTLS 1.2 [RFC6347] when this RFC was published. In the absence of a revision to this document, the latter requirement applies to all future versions of DTLS when they are published as RFCs. This document will only be revised if a revision to DTLS or SCTP makes a revision to the encapsulation necessary.

[NOTE to RFC-Editor:

Please ensure that the authors double check the above statement about DTLS 1.2 during AUTH48 and then remove this note before publication.

]

SCTP performs segmentation and reassembly based on the path MTU. Therefore the DTLS layer MUST NOT use any compression algorithm.

The DTLS MUST support sending messages larger than the current path MTU. This might result in sending IP level fragmented messages.

If path MTU discovery is performed by the DTLS layer, the method described in [RFC4821] MUST be used. For probe packets, the extension defined in [RFC6520] MUST be used.

If path MTU discovery is performed by the SCTP layer and IPv4 is used as the network layer protocol, the DTLS implementation SHOULD allow the DTLS user to enforce that the corresponding IPv4 packet is sent with the Don't Fragment (DF) bit set. If controlling the DF bit is not possible, for example due to implementation restrictions, a safe value for the path MTU has to be used by the SCTP stack. It is RECOMMENDED that the safe value does not exceed 1200 bytes. Please note that [RFC1122] only requires end hosts to be able to reassemble fragmented IP packets up to 576 bytes in length.

The DTLS implementation SHOULD allow the DTLS user to set the Differentiated services code point (DSCP) used for IP packets being sent (see [RFC2474]). This requires the DTLS implementation to pass the value through and the lower layer to allow setting this value. If the lower layer does not support setting the DSCP, then the DTLS user will end up with the default value used by protocol stack. Please note that only a single DSCP value can be used for all packets belonging to the same SCTP association.

Using explicit congestion notifications (ECN) in SCTP requires the DTLS layer to pass the ECN bits through and its lower layer to expose access to them for sent and received packets (see [RFC3168]). The implementation of DTLS and its lower layer have to provide this support. If this is not possible, for example due to implementation restrictions, ECN can't be used by SCTP.

6. SCTP Considerations

This section describes the usage of the base protocol and the applicability of various SCTP extensions.

6.1. Base Protocol

This document uses SCTP [RFC4960] with the following restrictions, which are required to reflect that the lower layer is DTLS instead of IPv4 and IPv6 and that SCTP does not deal with the IP addresses or the transport protocol used below DTLS:

- o A DTLS connection MUST be established before an SCTP association can be set up.
- o Multiple SCTP associations MAY be multiplexed over a single DTLS connection. The SCTP port numbers are used for multiplexing and demultiplexing the SCTP associations carried over a single DTLS connection.
- o All SCTP associations are single-homed, because DTLS does not expose any address management to its upper layer. Therefore it is RECOMMENDED to set the SCTP parameter `path.max.retrans` to `association.max.retrans`.
- o The INIT and INIT-ACK chunk MUST NOT contain any IPv4 Address or IPv6 Address parameters. The INIT chunk MUST NOT contain the Supported Address Types parameter.
- o The implementation MUST NOT rely on processing ICMP or ICMPv6 packets, since the SCTP layer most likely is unable to access the SCTP common header in the plain text of the packet, which triggered the sending of the ICMP or ICMPv6 packet. This applies in particular to path MTU discovery when performed by SCTP.
- o If the SCTP layer is notified about a path change by its lower layers, SCTP SHOULD retest the Path MTU and reset the congestion state to the initial state. The window-based congestion control method specified in [RFC4960], resets the congestion window and slow start threshold to their initial values.

6.2. Padding Extension

When the SCTP layer performs path MTU discovery as specified in [RFC4821], the padding extension defined in [RFC4820] MUST be supported and used for probe packets (HEARTBEAT chunks bundled with PADDING chunks [RFC4820]).

6.3. Dynamic Address Reconfiguration Extension

If the dynamic address reconfiguration extension defined in [RFC5061] is used, ASCONF chunks MUST use wildcard addresses only.

6.4. SCTP Authentication Extension

The SCTP authentication extension defined in [RFC4895] can be used with DTLS encapsulation, but does not provide any additional benefit.

6.5. Partial Reliability Extension

Partial reliability as defined in [RFC3758] can be used in combination with DTLS encapsulation. It is also possible to use additional PR-SCTP policies, for example the ones defined in [I-D.ietf-tsvwg-sctp-prpolicies].

6.6. Stream Reset Extension

The SCTP stream reset extension defined in [RFC6525] can be used with DTLS encapsulation. It is used to reset SCTP streams and add SCTP streams during the lifetime of the SCTP association.

6.7. Interleaving of Large User Messages

SCTP as defined in [RFC4960] does not support the interleaving of large user messages that need to be fragmented and reassembled by the SCTP layer. The protocol extension defined in [I-D.ietf-tsvwg-sctp-ndata] overcomes this limitation and can be used with DTLS encapsulation.

7. IANA Considerations

This document requires no actions from IANA.

8. Security Considerations

Security considerations for DTLS are specified in [RFC4347] and for SCTP in [RFC4960], [RFC3758], and [RFC6525]. The combination of SCTP and DTLS introduces no new security considerations.

SCTP should not process the IP addresses used for the underlying communication since DTLS provides no guarantees about them.

It should be noted that the inability to process ICMP or ICMPv6 messages does not add any security issue. When SCTP is carried over a connection-less lower layer like IPv4, IPv6, or UDP, processing of these messages is required to protect other nodes not supporting SCTP. Since DTLS provides a connection-oriented lower layer, this kind of protection is not necessary.

9. Acknowledgments

The authors wish to thank David Black, Benoit Claise, Spencer Dawkins, Francis Dupont, Gorrry Fairhurst, Stephen Farrell, Christer Holmberg, Barry Leiba, Eric Rescorla, Tom Taylor, Joe Touch and Magnus Westerlund for their invaluable comments.

10. References

10.1. Normative References

- [RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.
- [RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, March 2007.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.
- [RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.

10.2. Informative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, August 2007.
- [RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, February 2012.
- [RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013.
- [I-D.ietf-rtcweb-overview] Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-13 (work in progress), November 2014.

[I-D.ietf-rtcweb-data-channel]

Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ietf-tsvwg-sctp-prpolicies]

Tuexen, M., Seggelmann, R., Stewart, R., and S. Loreto, "Additional Policies for the Partial Reliability Extension of the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-prpolicies-06 (work in progress), December 2014.

[I-D.ietf-tsvwg-sctp-ndata]

Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "Stream Schedulers and a New Data Chunk for the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-ndata-02 (work in progress), January 2015.

Appendix A. NOTE to the RFC-Editor

Although the references to [I-D.ietf-tsvwg-sctp-prpolicies] and [I-D.ietf-tsvwg-sctp-ndata] are informative, put this document in REF-HOLD until these two references have been approved and update these references to the corresponding RFCs.

Authors' Addresses

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
DE

Email: tuexen@fh-muenster.de

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
US

Email: randall@lakerest.net

Randell Jesup
WorldGate Communications
3800 Horizon Blvd, Suite #103
Trevose, PA 19053-4947
US

Phone: +1-215-354-5166
Email: randell_ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: Salvatore.Loreto@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 20, 2016

Y. Nishida
GE Global Research
P. Natarajan
Cisco Systems
A. Caro
BBN Technologies
P. Amer
University of Delaware
K. Nielsen
Ericsson
February 17, 2016

SCTP-PF: Quick Failover Algorithm in SCTP
draft-ietf-tsvwg-sctp-failover-16.txt

Abstract

SCTP supports multi-homing. However, when the failover operation specified in RFC4960 is followed, there can be significant delay and performance degradation in the data transfer path failover. To overcome this problem this document specifies a quick failover algorithm (SCTP-PF) based on the introduction of a Potentially Failed (PF) state in SCTP Path Management.

The document also specifies a dormant state operation of SCTP. This dormant state operation is required to be followed by an SCTP-PF implementation, but it may equally well be applied by a standard RFC4960 SCTP implementation.

Additionally, the document introduces an alternative switchback operation mode called Primary Path Switchover that will be beneficial in certain situations. This mode of operation applies to both a standard RFC4960 SCTP implementation as well as to a SCTP-PF implementation.

The procedures defined in the document require only minimal modifications to the RFC4960 specification. The procedures are sender-side only and do not impact the SCTP receiver.

Status of This Memo

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 August 20, 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
2. Conventions and Terminology	4
3. SCTP with Potentially Failed Destination State (SCTP-PF) . .	4
3.1. Overview	4
3.2. Specification of the SCTP-PF Procedures	5
4. Dormant State Operation	9
4.1. SCTP Dormant State Procedure	10
5. Primary Path Switchover	11
6. Suggested SCTP Protocol Parameter Values	12
7. Socket API Considerations	12
7.1. Support for the Potentially Failed Path State	13
7.2. Peer Address Thresholds (SCTP_PEER_ADDR_THLDS) Socket Option	14
7.3. Exposing the Potentially Failed Path State (SCTP_EXPOSE_POTENTIALLY_FAILED_STATE) Socket Option . .	15
8. Security Considerations	15
9. MIB Considerations	16
10. IANA Considerations	16
11. Acknowledgements	16
12. Proposed Change of Status (to be Deleted before Publication)	17
13. References	17

13.1. Normative References	17
13.2. Informative References	17
Appendix A. Discussions of Alternative Approaches	18
A.1. Reduce Path.Max.Retrans (PMR)	18
A.2. Adjust RTO related parameters	19
Appendix B. Discussions for Path Bouncing Effect	20
Appendix C. SCTP-PF for SCTP Single-homed Operation	20
Authors' Addresses	21

1. Introduction

The Stream Control Transmission Protocol (SCTP) specified in [RFC4960] supports multi-homing at the transport layer. SCTP's multi-homing features include failure detection and failover procedures to provide network interface redundancy and improved end-to-end fault tolerance. In SCTP's current failure detection procedure, the sender must experience Path.Max.Retrans (PMR) number of consecutive failed timer-based retransmissions on a destination address before detecting a path failure. Until detecting the path failure, the sender continues to transmit data on the failed path. The prolonged time in which [RFC4960] SCTP continues to use a failed path severely degrades the performance of the protocol. To address this problem, this document specifies a quick failover algorithm (SCTP-PF) based on the introduction of a new Potentially Failed (PF) path state in SCTP path management. The performance deficiencies of the [RFC4960] failover operation, and the improvements obtainable from the introduction of a Potentially Failed state in SCTP, were proposed and documented in [NATARAJAN09] for Concurrent Multipath Transfer SCTP [IYENGAR06].

While SCTP-PF can accelerate failover process and improve performance, the risks that an SCTP endpoint enters the dormant state where all destination addresses are inactive can be increased. [RFC4960] leaves the protocol operation during dormant state to implementations and encourages to avoid entering the state as much as possible by careful tuning of the Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) parameters. We specify a dormant state operation for SCTP-PF which makes SCTP-PF provide the same disruption tolerance as [RFC4960] despite that the dormant state may be entered more quickly. The dormant state operation may equally well be applied by an [RFC4960] implementation and will here serve to provide added fault tolerance for situations where the tuning of the Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) parameters fail to provide adequate prevention of the entering of the dormant state.

The operation after the recovery of a failed path also impacts the performance of the protocol. With the procedures specified in

[RFC4960] SCTP will, after a failover from the primary path, switch back to use the primary path for data transfer as soon as this path becomes available again. From a performance perspective such a forced switchback of the data transmission path can be suboptimal as the CWND towards the original primary destination address has to be rebuilt once data transfer resumes, [CARO02]. As an optional alternative to the switchback operation of [RFC4960], this document specifies an alternative Primary Path Switchover procedure which avoid such forced switchbacks of the data transfer path. The Primary Path Switchover operation was originally proposed in [CARO02].

While SCTP-PF primarily is motivated by a desire to improve the multi-homed operation, the feature applies also to SCTP single-homed operation. Here the algorithm serves to provide increased failure detection on idle associations, whereas the failover or switchback aspects of the algorithm will not be activated. This is discussed in more detail in Appendix C.

A brief description of the motivation for the introduction of the Potentially Failed state including a discussion of alternative approaches to mitigate the deficiencies of the [RFC4960] failover operation are given in the Appendices. Discussion of path bouncing effects that might be caused by frequent switchovers, are also provided there.

2. Conventions and Terminology

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

3. SCTP with Potentially Failed Destination State (SCTP-PF)

3.1. Overview

To minimize the performance impact during failover, the sender should avoid transmitting data to a failed destination address as early as possible. In the [RFC4960] SCTP path management scheme, the sender stops transmitting data to a destination address only after the destination address is marked inactive. This process takes a significant amount of time as it requires the error counter of the destination address to exceed the Path.Max.Retrans (PMR) threshold. The issue cannot simply be mitigated by lowering of the PMR threshold because this may result in spurious failure detection and unnecessary prevention of the usage of a preferred primary path. Also due to the coupled tuning of the Path.Max.Retrans (PMR) and the Association.Max.Retrans (AMR) parameter values in [RFC4960], lowering

of the PMR threshold may result in lowering of the AMR threshold, which would result in decrease of the fault tolerance of SCTP.

The solution provided in this document is to extend the SCTP path management scheme of [RFC4960] by the addition of the Potentially Failed (PF) state as an intermediate state in between the active and inactive state of a destination address in the [RFC4960] path management scheme, and let the failover of data transfer away from a destination address be driven by the entering of the PF state instead of by the entering of the inactive state. Thereby SCTP may perform quick failover without negatively impacting the overall fault tolerance of [RFC4960] SCTP. At the same time, RTO-based HEARTBEAT probing is initiated towards a destination address once it enters PF state. Thereby SCTP may quickly ascertain whether network connectivity towards the destination address is broken or whether the failover was spurious. In the case where the failover was spurious data transfer may quickly resume towards the original destination address.

The new failure detection algorithm assumes that loss detected by a timeout implies either severe congestion or network connectivity failure. It recommends that by default a destination address is classified as PF at the occurrence of the first timeout.

3.2. Specification of the SCTP-PF Procedures

The SCTP-PF operation is specified as follows:

1. The sender maintains a new tunable SCTP Protocol Parameter called PotentiallyFailed.Max.Retrans (PFMR). The PFMR defines the new intermediate PF threshold on the destination address error counter. When this threshold is exceeded the destination address is classified as PF. The RECOMMENDED value of PFMR is 0. If PFMR is set to be greater than or equal to Path.Max.Retrans (PMR), the resulting PF threshold will be so high that the destination address will reach the inactive state before it can be classified as PF.
2. The error counter of an active destination address is incremented or cleared as specified in [RFC4960]. This means that the error counter of the destination address in active state will be incremented each time the T3-rtx timer expires, or each time a HEARTBEAT chunk is sent when idle and not acknowledged within an RTO. When the value in the destination address error counter exceeds PFMR, the endpoint MUST mark the destination address as in the PF state.

3. A SCTP-PF sender SHOULD NOT send data to destination addresses in PF state when alternative destination addresses in active state are available. Specifically this means that:
 - i When there is outbound data to send and the destination address presently used for data transmission is in PF state, the sender SHOULD choose a destination address in active state, if one exists, and use this destination address for data transmission.
 - ii As specified in [RFC4960] section 6.4.1, when the sender retransmits data that has timed out, it should attempt to pick a new destination address for data retransmission. In this case, the sender SHOULD choose an alternate destination transport address in active state if one exists.
 - iii When there is outbound data to send and the SCTP user explicitly requests to send data to a destination address in PF state, the sender SHOULD send the data to an alternate destination address in active state if one exists.

When choosing among multiple destination addresses in active state an SCTP sender will follow the guiding principles of section 6.4.1 of [RFC4960] of choosing most divergent source-destination pairs compared with, for i.: the destination address in PF state that it performs a failover from, and for ii.: the destination address towards which the data timed out. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document.

In all cases, the sender MUST NOT change the state of chosen destination address, whether this state be active or PF, and it MUST NOT clear the error counter of the destination address as a result of choosing the destination address for data transmission.

4. When the destination addresses are all in PF state or some in PF state and some in inactive state, the sender MUST choose one destination address in PF state and SHOULD transmit or retransmit data to this destination address using the following rules:
 - A. The sender SHOULD choose the destination in PF state with the lowest error count (fewest consecutive timeouts) for data transmission and transmit or retransmit data to this destination.

- B. When there are multiple destination addresses in PF state with same error count, the sender should let the choice among the multiple destination addresses in PF state with equal error count be based on the [RFC4960], section 6.4.1, principles of choosing most divergent source-destination pairs when executing (potentially consecutive) retransmission. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document.

The sender MUST NOT change the state and the error counter of any destination addresses as the result of the selection.

- 5. The HB.interval of the Path Heartbeat function of [RFC4960] MUST be ignored for destination addresses in PF state. Instead HEARTBEAT chunks are sent to destination addresses in PF state once per RTO. HEARTBEAT chunks SHOULD be sent to destination addresses in PF state, but the sending of HEARTBEATS MUST honor whether the Path Heartbeat function (Section 8.3 of [RFC4960]) is enabled for the destination address or not. I.e., if the Path Heartbeat function is disabled for the destination address in question, HEARTBEATS MUST NOT be sent. Note that when Heartbeat function is disabled, it may take longer to transition a destination address in PF state back to active state.
- 6. HEARTBEATS are sent when a destination address reaches the PF state. When a HEARTBEAT chunk is not acknowledged within the RTO, the sender increments the error counter and exponentially backs off the RTO value. If the error counter is less than PMR, the sender transmits another packet containing the HEARTBEAT chunk immediately after timeout expiration on the previous HEARTBEAT. When data is being transmitted to a destination address in the PF state, the transmission of a HEARTBEAT chunk MAY be omitted in case where the receipt of a SACK of the data or a T3-rtx timer expiration on the data can provide equivalent information, such as the case where the data chunk has been transmitted to a single destination address only. Likewise, the timeout of a HEARTBEAT chunk MAY be ignored if data is outstanding towards the destination address.
- 7. When the sender receives a HEARTBEAT ACK from a HEARTBEAT sent to a destination address in PF state, the sender SHOULD clear the error counter of the destination address and transition the destination address back to active state. However, there may be a situation where HEARTBEAT chunks can go through while DATA chunks cannot. Hence, in a situation where a HEARTBEAT ACK arrives while there is data outstanding towards the destination address to which the HEARTBEAT was sent, then an implementation

MAY choose to not have the HEARTBEAT ACK reset the error counter, but have the error counter reset await the fate of the outstanding data transmission. This situation can happen when data is sent to a destination address in PF state. When the sender resumes data transmission on a destination address after a transition of the destination address from PF to active state, it MUST do this following the prescriptions of Section 7.2 of [RFC4960].

8. Additional (PMR - PFMR) consecutive timeouts on a destination address in PF state confirm the path failure, upon which the destination address transitions to the inactive state. As described in [RFC4960], the sender (i) SHOULD notify the ULP about this state transition, and (ii) transmit HEARTBEAT chunks to the inactive destination address at a lower HB.interval frequency as described in Section 8.3 of [RFC4960] (when the Path Heartbeat function is enabled for the destination address).
9. Acknowledgments for chunks that have been transmitted to multiple destinations (i.e., a chunk which has been retransmitted to a different destination address than the destination address to which the chunk was first transmitted) SHOULD NOT clear the error count for an inactive destination address and SHOULD NOT move a destination address in PF state back to active state, since a sender cannot disambiguate whether the ACK was for the original transmission or the retransmission(s). A SCTP sender MAY clear the error counter and move a destination address back to active state by information other than acknowledgments, when it can uniquely determine which destination, among multiple destination addresses, the chunk reached. This document makes no reference to what such information could consist of, nor how such information could be obtained.
10. Acknowledgments for data chunks that has been transmitted to one destination address only MUST clear the error counter for the destination address and MUST transition a destination address in PF state back to active state. This situation can happen when new data is sent to a destination address in the PF state. It can also happen in situations where the destination address is in the PF state due to the occurrence of a spurious T3-rtx timer and acknowledgments start to arrive for data sent prior to occurrence of the spurious T3-rtx and data has not yet been retransmitted towards other destinations. This document does not specify special handling for detection of or reaction to spurious T3-rtx timeouts, e.g., for special operation vis-a-vis the congestion control handling or data retransmission operation towards a destination address which undergoes a transition from

active to PF to active state due to a spurious T3-rtx timeout. But it is noted that this is an area which would benefit from additional attention, experimentation and specification for single-homed SCTP as well as for multi-homed SCTP protocol operation.

11. When all destination addresses are in inactive state, and SCTP protocol operation thus is said to be in dormant state, the prescriptions given in Section 4 shall be followed.
12. The SCTP stack SHOULD expose the PF state of its destination addresses to the ULP as well as provide the means to notify the ULP of state transitions of its destination addresses from active to PF, and vice-versa. However it is recommended that an SCTP stack implementing SCTP-PF also allows for that the ULP is kept ignorant of the PF state of its destinations and the associated state transitions, thus allowing for retain of the simpler state transition model of RFC4960 in the ULP. For this reason it is recommended that an SCTP stack implementing SCTP-PF also provides the ULP with the means to suppress exposure of the PF state and the associated state transitions.

4. Dormant State Operation

In a situation with complete disruption of the communication in between the SCTP Endpoints, the aggressive HEARTBEAT transmissions of SCTP-PF on destination addresses in PF state may make the association enter dormant state faster than a standard [RFC4960] SCTP implementation given the same setting of Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR). For example, an SCTP association with two destination addresses typically would reach dormant state in half the time of an [RFC4960] SCTP implementation in such situations. This is because a SCTP PF sender will send HEARTBEATS and data retransmissions in parallel with RTO intervals when there are multiple destinations addresses in PF state. This argument presumes that $RTO \ll HB.interval$ of [RFC4960]. With the design goal that SCTP-PF shall provide the same level of disruption tolerance as an [RFC4960] SCTP implementation with the same Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) setting, we prescribe for that an SCTP-PF implementation SHOULD operate as described below in Section 4.1 during dormant state.

An SCTP-PF implementation MAY choose a different dormant state operation than the one described below in Section 4.1 provided that the solution chosen does not decrease the fault tolerance of the SCTP-PF operation.

The below prescription for SCTP-PF dormant state handling MUST NOT be coupled to the value of the PFMR, but solely to the activation of SCTP-PF logic in an SCTP implementation.

It is noted that the below dormant state operation is considered to provide added disruption tolerance also for an [RFC4960] SCTP implementation, and that it can be sensible for an [RFC4960] SCTP implementation to follow this mode of operation. For an [RFC4960] SCTP implementation the continuation of data transmission during dormant state makes the fault tolerance of SCTP be more robust towards situations where some, or all, alternative paths of an SCTP association approach, or reach, inactive state before the primary path used for data transmission observes trouble.

4.1. SCTP Dormant State Procedure

- a. When the destination addresses are all in inactive state and data is available for transfer, the sender MUST choose one destination and transmit data to this destination address.
- b. The sender MUST NOT change the state of the chosen destination address (it remains in inactive state) and it MUST NOT clear the error counter of the destination address as a result of choosing the destination address for data transmission.
- c. The sender SHOULD choose the destination in inactive state with the lowest error count (fewest consecutive timeouts) for data transmission. When there are multiple destinations with same error count in inactive state, the sender SHOULD attempt to pick the most divergent source - destination pair from the last source - destination pair where failure was observed. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document. To support differentiation of inactive destination addresses based on their error count SCTP will need to allow for increment of the destination address error counters up to some reasonable limit above PMR+1, thus changing the prescriptions of [RFC4960], section 8.3, in this respect. The exact limit to apply is not specified in this document but it is considered reasonable to require for the limit to be an order of magnitude higher than the PMR value. A sender MAY choose to deploy other strategies than the strategy defined here. The strategy to prioritize the last active destination address, i.e., the destination address with the fewest error counts is optimal when some paths are permanently inactive, but suboptimal when a path instability is transient.

5. Primary Path Switchover

The objective of the Primary Path Switchover operation is to allow the SCTP sender to continue data transmission on a new working path even when the old primary destination address becomes active again. This is achieved by having SCTP perform a switchover of the primary path to the new working path if the error counter of the primary path exceeds a certain threshold. This mode of operation can be applied not only to SCTP-PF implementations, but also to [RFC4960] implementations.

The Primary Path Switchover operation requires only sender side changes. The details are:

1. The sender maintains a new tunable parameter, called Primary.Switchover.Max.Retrans (PSMR). For SCTP-PF implementations, the PSMR MUST be set greater or equal to the PFMR value. For [RFC4960] implementations the PSMR MUST be set greater or equal to the PMR value. Implementations MUST reject any other values of PSMR.
2. When the path error counter on a set primary path exceeds PSMR, the SCTP implementation MUST autonomously select and set a new primary path.
3. The primary path selected by the SCTP implementation MUST be the path which at the given time would be chosen for data transfer. A previously failed primary path can be used as data transfer path as per normal path selection when the present data transfer path fails.
4. For SCTP-PF, the recommended value of PSMR is PFMR when Primary Path Switchover operation mode is used. This means that no forced switchback to a previously failed primary path is performed. An SCTP-PF implementation of Primary Path Switchover MUST support the setting of PSMR = PFMR. A SCTP-PF implementation of Primary Path Switchover MAY support setting of PSMR > PFMR.
5. For [RFC4960] SCTP, the recommended value of PSMR is PMR when Primary Path Switchover is used. This means that no forced switchback to a previously failed primary path is performed. A [RFC4960] SCTP implementation of Primary Path Switchover MUST support the setting of PSMR = PMR. An [RFC4960] SCTP implementation of Primary Path Switchover MAY support larger settings of PSMR > PMR.

6. It MUST be possible to disable the Primary Path Switchover operation and obtain the standard switchback operation of [RFC4960].

The manner of switchover operation that is most optimal in a given scenario depends on the relative quality of a set primary path versus the quality of alternative paths available as well as on the extent to which it is desired for the mode of operation to enforce traffic distribution over a number of network paths. I.e., load distribution of traffic from multiple SCTP associations may be sought to be enforced by distribution of the set primary paths with [RFC4960] switchback operation. However as [RFC4960] switchback behavior is suboptimal in certain situations, especially in scenarios where a number of equally good paths are available, an SCTP implementation MAY support also, as alternative behavior, the Primary Path Switchover mode of operation and MAY enable it based on applications' requests.

For an SCTP implementation that implements the Primary Path Switchover operation, this specification RECOMMENDS that the standard RFC4960 switchback operation is retained as the default operation.

6. Suggested SCTP Protocol Parameter Values

This document does not alter the [RFC4960] value recommendation for the SCTP Protocol Parameters defined in [RFC4960].

The following protocol parameter is RECOMMENDED:

PotentiallyFailed.Max.Retrans (PFMR) - 0

7. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to control and observe the SCTP-PF behavior as well as the Primary Path Switchover function.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is, by means of the existing SCTP_PEER_ADDR_CHANGE event, extended to provide the event notification when a peer address enters or leaves the potentially failed state as well as the socket API implementation is extended to expose the potentially failed state of a peer address in the existing SCTP_GET_PEER_ADDR_INFO structure.

Furthermore, two new read/write socket options for the level IPPROTO_SCTP and the name SCTP_PEER_ADDR_THLDS and

SCTP_EXPOSE_POTENTIALLY_FAILED_STATE are defined as described below. The first socket option is used to control the values of the PFMR and PSMP parameters described in Section 3 and in Section 5. The second one controls the exposition of the potentially failed path state.

Support for the SCTP_PEER_ADDR_THLDS and SCTP_EXPOSE_POTENTIALLY_FAILED_STATE socket options need also to be added to the function sctp_opt_info().

7.1. Support for the Potentially Failed Path State

As defined in [RFC6458], the SCTP_PEER_ADDR_CHANGE event is provided if the status of a peer address changes. In addition to the state changes described in [RFC6458], this event is also provided, if a peer address enters or leaves the potentially failed state. The notification as defined in [RFC6458] uses the following structure:

```
struct sctp_paddr_change {
    uint16_t spc_type;
    uint16_t spc_flags;
    uint32_t spc_length;
    struct sockaddr_storage spc_aaddr;
    uint32_t spc_state;
    uint32_t spc_error;
    sctp_assoc_t spc_assoc_id;
}
```

[RFC6458] defines the constants SCTP_ADDR_AVAILABLE, SCTP_ADDR_UNREACHABLE, SCTP_ADDR_REMOVED, SCTP_ADDR_ADDED, and SCTP_ADDR_MADE_PRIM to be provided in the spc_state field. This document defines in addition to that the new constant SCTP_ADDR_POTENTIALLY_FAILED, which is reported if the affected address becomes potentially failed.

The SCTP_GET_PEER_ADDR_INFO socket option defined in [RFC6458] can be used to query the state of a peer address. It uses the following structure:

```
struct sctp_paddrinfo {
    sctp_assoc_t spinfo_assoc_id;
    struct sockaddr_storage spinfo_address;
    int32_t spinfo_state;
    uint32_t spinfo_cwnd;
    uint32_t spinfo_srtt;
    uint32_t spinfo_rto;
    uint32_t spinfo_mtu;
};
```

[RFC6458] defines the constants `SCTP_UNCONFIRMED`, `SCTP_ACTIVE`, and `SCTP_INACTIVE` to be provided in the `spinfo_state` field. This document defines in addition to that the new constant `SCTP_POTENTIALLY_FAILED`, which is reported if the peer address is potentially failed.

7.2. Peer Address Thresholds (`SCTP_PEER_ADDR_THLDS`) Socket Option

Applications can control the SCTP-PF behavior by getting or setting the number of consecutive timeouts before a peer address is considered potentially failed or unreachable. The same socket option is used by applications to set and get the number of timeouts before the primary path is changed automatically by the Primary Path Switchover function. This socket option uses the level `IPPROTO_SCTP` and the name `SCTP_PEER_ADDR_THLDS`.

The following structure is used to access and modify the thresholds:

```
struct sctp_paddrthlds {
    sctp_assoc_t spt_assoc_id;
    struct sockaddr_storage spt_address;
    uint16_t spt_pathmaxrxt;
    uint16_t spt_pathpfthld;
    uint16_t spt_pathcpthld;
};
```

`spt_assoc_id`: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application may fill in an association identifier or `SCTP_FUTURE_ASSOC`. It is an error to use `SCTP_{CURRENT|ALL}_ASSOC` in `spt_assoc_id`.

`spt_address`: This specifies which peer address is of interest. If a wild card address is provided, this socket option applies to all current and future peer addresses.

`spt_pathmaxrxt`: Each peer address of interest is considered unreachable, if its path error counter exceeds `spt_pathmaxrxt`.

`spt_pathpfthld`: Each peer address of interest is considered Potentially Failed, if its path error counter exceeds `spt_pathpfthld`.

`spt_pathcpthld`: Each peer address of interest is not considered the primary remote address anymore, if its path error counter exceeds `spt_pathcpthld`. Using a value of `0xffff` disables the selection of a new primary peer address. If an implementation does not support the automatic selection of a new primary address, it should indicate an error with `errno` set to `EINVAL` if a value different

from 0xffff is used in `spt_pathcpthld`. For SCTP-PF, the setting of `spt_pathcpthld < spt_pathpfthld` should be rejected with `errno` set to `EINVAL`. For [RFC4960] SCTP, the setting of `spt_pathcpthld < spt_pathmaxrxt` should be rejected with `errno` set to `EINVAL`. A SCTP-PF implementation may support only setting of `spt_pathcpthld = spt_pathpfthld` and `spt_pathcpthld = 0xffff` and a [RFC4960] SCTP implementation may support only setting of `spt_pathcpthld = spt_pathmaxrxt` and `spt_pathcpthld = 0xffff`. In these cases SCTP shall reject setting of other values with `errno` set to `EINVAL`.

7.3. Exposing the Potentially Failed Path State (`SCTP_EXPOSE_POTENTIALLY_FAILED_STATE`) Socket Option

Applications can control the exposure of the potentially failed path state in the `SCTP_PEER_ADDR_CHANGE` event and the `SCTP_GET_PEER_ADDR_INFO` as described in Section 7.1. The default value is implementation specific.

This socket option uses the level `IPPROTO_SCTP` and the name `SCTP_EXPOSE_POTENTIALLY_FAILED_STATE`.

The following structure is used to control the exposition of the potentially failed path state:

```
struct sctp_assoc_value {
    sctp_assoc_t assoc_id;
    uint32_t assoc_value;
};
```

`assoc_id`: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application may fill in an association identifier or `SCTP_FUTURE_ASSOC`. It is an error to use `SCTP_{CURRENT|ALL}_ASSOC` in `assoc_id`.

`assoc_value`: The potentially failed path state is exposed if and only if this parameter is non-zero.

8. Security Considerations

Security considerations for the use of SCTP and its APIs are discussed in [RFC4960] and [RFC6458].

The logic introduced by this document does not impact existing SCTP messages on the wire. Also, this document does not introduce any new SCTP messages on the wire that require new security considerations.

SCTP-PF makes SCTP not only more robust during primary path failure/congestion but also more vulnerable to network connectivity/

congestion attacks on the primary path. SCTP-PF makes it easier for an attacker to trick SCTP to change data transfer path, since the duration of time that an attacker needs to negatively influence the network connectivity is much shorter than [RFC4960]. However, SCTP-PF does not constitute a significant change in the duration of time and effort an attacker needs to keep SCTP away from the primary path. With the standard switchback operation [RFC4960] SCTP resumes data transfer on its primary path as soon as the next HEARTBEAT succeeds.

On the other hand, usage of the Primary Path Switchover mechanism, does change the threat analysis. This is because on-path attackers can force a permanent change of the data transfer path by blocking the primary path until the switchover of the primary path is triggered by the Primary Path Switchover algorithm. This especially will be the case when the Primary Path Switchover is used together with SCTP-PF with the particular setting of PSMR = PFMR = 0, as Primary Path Switchover here happens already at the first RTO timeout experienced. Users of the Primary Path Switchover mechanism should be aware of this fact.

The event notification of path state transfer from active to potentially failed state and vice versa gives attackers an increased possibility to generate more local events. However, it is assumed that event notifications are rate-limited in the implementation to address this threat.

9. MIB Considerations

SCTP-PF introduces new SCTP algorithms for failover and switchback with associated new state parameters. It is recommended that the SCTP-MIB defined in [RFC3873] is updated to support the management of the SCTP-PF implementation. This can be done by extending the sctpAssocRemAddrActive field of the SCTPAssocRemAddrTable to include information of the PF state of the destination address and by adding new fields to the SCTPAssocRemAddrTable supporting PotentiallyFailed.Max.Retrans (PFMR) and Primary.Switchover.Max.Retrans (PSMR) parameters.

10. IANA Considerations

This document does not create any new registries or modify the rules for any existing registries managed by IANA.

11. Acknowledgements

The authors wish to thank Michael Tuexen for his many invaluable comments and for his very substantial support with the making of this document.

12. Proposed Change of Status (to be Deleted before Publication)

Initially this work looked to entail some changes of the Congestion Control (CC) operation of SCTP and for this reason the work was proposed as Experimental. These intended changes of the CC operation have since been judged to be irrelevant and are no longer part of the specification. As the specification entails no other potential harmful features, consensus exists in the WG to bring the work forward as PS.

Initially concerns have been expressed about the possibility for the mechanism to introduce path bouncing with potential harmful network impacts. These concerns are believed to be unfounded. This issue is addressed in Appendix B.

It is noted that the feature specified by this document is implemented by multiple SCTP SW implementations and furthermore that various variants of the solution have been deployed in telephony signaling environments for several years with good results.

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

13.2. Informative References

- [CARO02] Caro Jr., A., Iyengar, J., Amer, P., Heinz, G., and R. Stewart, "A Two-level Threshold Recovery Mechanism for SCTP", Tech report, CIS Dept, University of Delaware , 7 2002.
- [CARO04] Caro Jr., A., Amer, P., and R. Stewart, "End-to-End Failover Thresholds for Transport Layer Multi homing", MILCOM 2004 , 11 2004.
- [CARO05] Caro Jr., A., "End-to-End Fault Tolerance using Transport Layer Multi homing", Ph.D Thesis, University of Delaware , 1 2005.

- [FALLON08] Fallon, S., Jacob, P., Qiao, Y., Murphy, L., Fallon, E., and A. Hanley, "SCTP Switchover Performance Issues in WLAN Environments", IEEE CCNC 2008, 1 2008.
- [GRINNEMO04] Grinnemo, K-J. and A. Brunstrom, "Performance of SCTP-controlled failovers in M3UA-based SIGTRAN networks", Advanced Simulation Technologies Conference , 4 2004.
- [IYENGAR06] Iyengar, J., Amer, P., and R. Stewart, "Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-end Paths.", IEEE/ACM Trans on Networking 14(5), 10 2006.
- [JUNGMAIER02] Jungmaier, A., Rathgeb, E., and M. Tuexen, "On the use of SCTP in failover scenarios", World Multiconference on Systemics, Cybernetics and Informatics , 7 2002.
- [NATARAJAN09] Natarajan, P., Ekiz, N., Amer, P., and R. Stewart, "Concurrent Multipath Transfer during Path Failure", Computer Communications , 5 2009.
- [RFC3873] Pastor, J. and M. Belinchon, "Stream Control Transmission Protocol (SCTP) Management Information Base (MIB)", RFC 3873, DOI 10.17487/RFC3873, September 2004, <<http://www.rfc-editor.org/info/rfc3873>>.
- [RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, December 2011.

Appendix A. Discussions of Alternative Approaches

This section lists alternative approaches for the issues described in this document. Although these approaches do not require to update RFC4960, we do not recommend them from the reasons described below.

A.1. Reduce Path.Max.Retrans (PMR)

Smaller values for Path.Max.Retrans shorten the failover duration and in fact this is recommended in some research results [JUNGMAIER02] [GRINNEMO04] [FALLON08]. However to significantly reduce the failover time it is required to go down (as with PFMR) to Path.Max.Retrans=0 and with this setting SCTP switches to another

destination address already on a single timeout which may result in spurious failover. Spurious failover is a problem in [RFC4960] SCTP as the transmission of HEARTBEATS on the left primary path, unlike in SCTP-PF, is governed by 'HB.interval' also during the failover process. 'HB.interval' is usually set in the order of seconds (recommended value is 30 seconds) and when the primary path becomes inactive, the next HEARTBEAT may be transmitted only many seconds later. Indeed as recommended, only 30 secs later. Meanwhile, the primary path may since long have recovered, if it needed recovery at all (indeed the failover could be truly spurious). In such situations, post failover, an endpoint is forced to wait in the order of many seconds before the endpoint can resume transmission on the primary path and furthermore once it returns on the primary path the CWND needs to be rebuild anew - a process which the throughput already have had to suffer from on the alternate path. Using a smaller value for 'HB.interval' might help this situation, but it would result in a general waste of bandwidth as such more frequent HEARTBEATING would take place also when there are no observed troubles. The bandwidth overhead may be diminished by having the ULP use a smaller 'HB.interval' only on the path which at any given time is set to be the primary path, but this adds complication in the ULP.

In addition, smaller Path.Max.Retrans values also affect the 'Association.Max.Retrans' value. When the SCTP association's error count exceeds Association.Max.Retrans threshold, the SCTP sender considers the peer endpoint unreachable and terminates the association. Section 8.2 in [RFC4960] recommends that Association.Max.Retrans value should not be larger than the summation of the Path.Max.Retrans of each of the destination addresses. Else the SCTP sender considers its peer reachable even when all destinations are INACTIVE and to avoid this dormant state operation, [RFC4960] SCTP implementation SHOULD reduce Association.Max.Retrans accordingly whenever it reduces Path.Max.Retrans. However, smaller Association.Max.Retrans value decreases the fault tolerance of SCTP as it increases the chances of association termination during minor congestion events.

A.2. Adjust RTO related parameters

As several research results indicate, we can also shorten the duration of failover process by adjusting RTO related parameters [JUNGMAIER02] [FALLON08]. During failover process, RTO keeps being doubled. However, if we can choose smaller value for RTO.max, we can stop the exponential growth of RTO at some point. Also, choosing smaller values for RTO.initial or RTO.min can contribute to keep the RTO value small.

Similar to reducing Path.Max.Retrans, the advantage of this approach is that it requires no modification to the current specification, although it needs to ignore several recommendations described in the Section 15 of [RFC4960]. However, this approach requires to have enough knowledge about the network characteristics between end points. Otherwise, it can introduce adverse side-effects such as spurious timeouts.

The significant issue with this approach, however, is that even if the RTO.max is lowered to an optimal low value, then as long as the Path.Max.Retrans is kept at the [RFC4960] recommended value, the reduction of the RTO.max doesn't reduce the failover time sufficiently enough to prevent severe performance degradation during failover.

Appendix B. Discussions for Path Bouncing Effect

The methods described in the document can accelerate the failover process. Hence, they might introduce the path bouncing effect where the sender keeps changing the data transmission path frequently. This sounds harmful to the data transfer, however several research results indicate that there is no serious problem with SCTP in terms of path bouncing effect [CARO04] [CARO05].

There are two main reasons for this. First, SCTP is basically designed for multipath communication, which means SCTP maintains all path related parameters (CWND, ssthresh, RTT, error count, etc) per each destination address. These parameters cannot be affected by path bouncing. In addition, when SCTP migrates the data transfer to another path, it starts with the minimal or the initial CWND. Hence, there is little chance for packet reordering or duplicating.

Second, even if all communication paths between the end-nodes share the same bottleneck, the SCTP-PF results in a behavior already allowed by [RFC4960].

Appendix C. SCTP-PF for SCTP Single-homed Operation

For a single-homed SCTP association the only tangible effect of the activation of SCTP-PF operation is enhanced failure detection in terms of potential notification of the PF state of the sole destination address as well as, for idle associations, more rapid entering, and notification, of inactive state of the destination address and more rapid end-point failure detection. It is believed that neither of these effects are harmful, provided adequate dormant state operation is implemented, and furthermore that they may be particularly useful for applications that deploys multiple SCTP associations for load balancing purposes. The early notification of

the PF state may be used for preventive measures as the entering of the PF state can be used as a warning of potential congestion. Depending on the PMR value, the aggressive HEARTBEAT transmission in PF state may speed up the end-point failure detection (exceed of AMR threshold on the sole path error counter) on idle associations in case where relatively large HB.interval value compared to RTO (e.g. 30secs) is used.

Authors' Addresses

Yoshifumi Nishida
GE Global Research
2623 Camino Ramon
San Ramon, CA 94583
USA

Email: nishida@wide.ad.jp

Preethi Natarajan
Cisco Systems
510 McCarthy Blvd
Milpitas, CA 95035
USA

Email: prenatar@cisco.com

Armando Caro
BBN Technologies
10 Moulton St.
Cambridge, MA 02138
USA

Email: acaro@bbn.com

Paul D. Amer
University of Delaware
Computer Science Department - 434 Smith Hall
Newark, DE 19716-2586
USA

Email: amer@udel.edu

Karen E. E. Nielsen
Ericsson
Kistavaegen 25
Stockholm 164 80
Sweden

Email: karen.nielsen@tieto.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 5, 2018

R. Stewart
Netflix, Inc.
M. Tuexen
Muenster Univ. of Appl. Sciences
S. Loreto
Ericsson
R. Seggelmann
Metafinanz Informationssysteme GmbH
September 1, 2017

Stream Schedulers and User Message Interleaving for the Stream Control
Transmission Protocol
draft-ietf-tsvwg-sctp-ndata-13.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a message oriented transport protocol supporting arbitrarily large user messages. This document adds a new chunk to SCTP for carrying payload data. This allows a sender to interleave different user messages that would otherwise result in head of line blocking at the sender. The interleaving of user messages is required for WebRTC Datachannels.

Whenever an SCTP sender is allowed to send user data, it may choose from multiple outgoing SCTP streams. Multiple ways for performing this selection, called stream schedulers, are defined in this document. A stream scheduler can choose to either implement, or not implement, user message interleaving.

Status of This Memo

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.

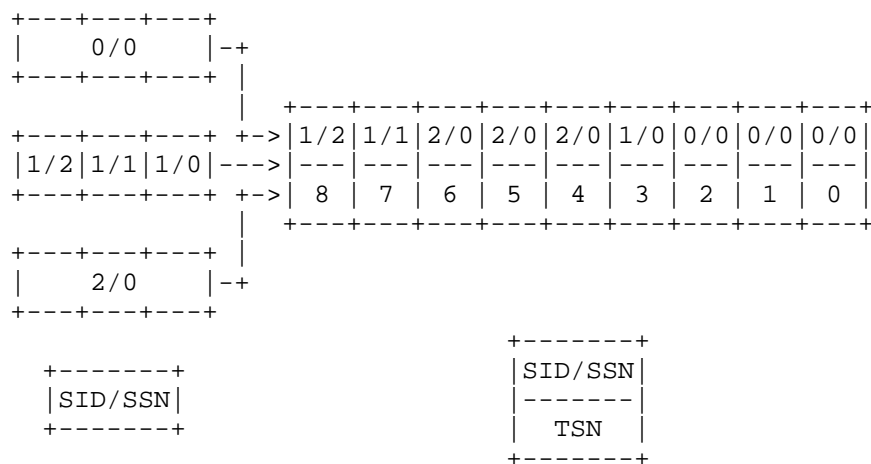


Figure 1: Round Robin Scheduler without User Message Interleaving

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.

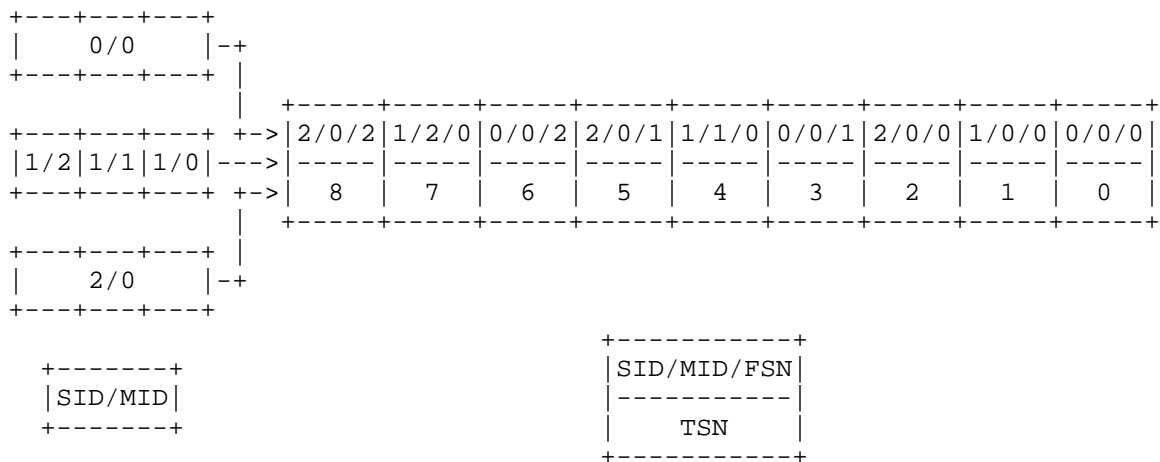


Figure 2: Round Robin Scheduler with User Message Interleaving

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.

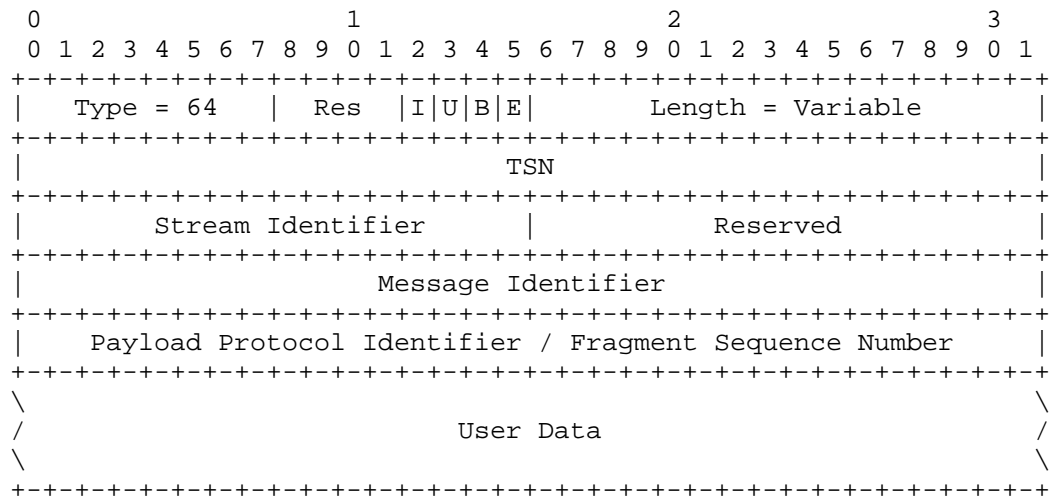


Figure 3: I-DATA chunk format

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.

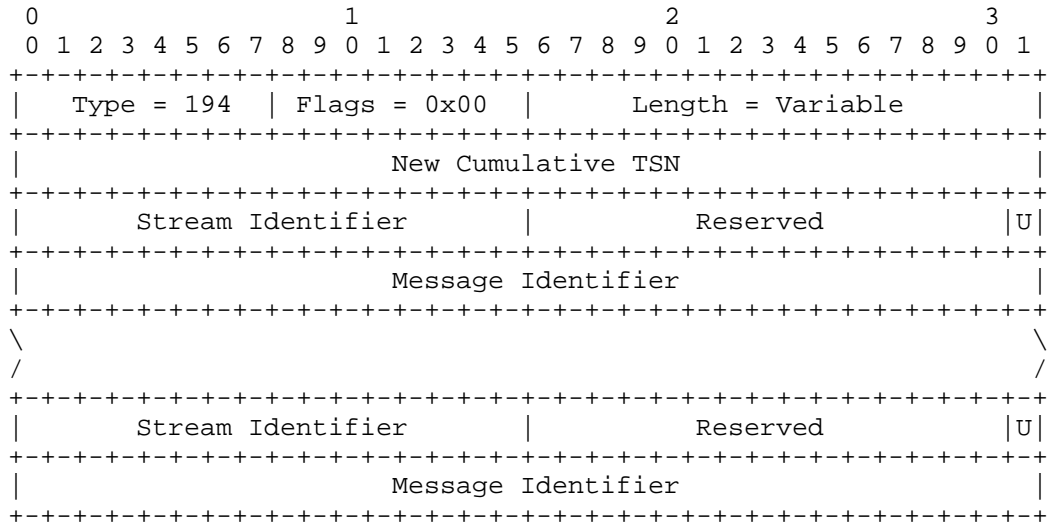


Figure 4: I-FORWARD-TSN chunk format

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 11, 2015

M. Tuexen
Muenster Univ. of Appl. Sciences
R. Seggelmann
T-Systems International GmbH
R. Stewart
Netflix, Inc.
S. Loreto
Ericsson
February 7, 2015

Additional Policies for the Partial Reliability Extension of the Stream
Control Transmission Protocol
draft-ietf-tsvwg-sctp-prpolicies-07.txt

Abstract

This document defines two additional policies for the Partial Reliability Extension of the Stream Control Transmission Protocol (PR-SCTP) allowing to limit the number of retransmissions or to prioritize user messages for more efficient send buffer 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 August 11, 2015.

Copyright Notice

Copyright (c) 2015 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	3
3. Additional PR-SCTP Policies	3
3.1. Limited Retransmissions Policy	3
3.2. Priority Policy	3
4. Socket API Considerations	4
4.1. Data Types	4
4.2. Support for Added PR-SCTP Policies	4
4.3. Socket Option for Getting the Stream Specific PR-SCTP Status (SCTP_PR_STREAM_STATUS)	5
4.4. Socket Option for Getting the Association Specific PR- SCTP Status (SCTP_PR_ASSOC_STATUS)	6
4.5. Socket Option for Getting and Setting the PR-SCTP Support (SCTP_PR_SUPPORTED)	7
5. IANA Considerations	8
6. Security Considerations	8
7. Acknowledgments	8
8. References	8
8.1. Normative References	8
8.2. Informative References	9
Authors' Addresses	9

1. Introduction

The SCTP Partial Reliability Extension (PR-SCTP) defined in [RFC3758] provides a generic method for senders to abandon user messages. The decision to abandon a user message is sender side only and the exact condition is called a PR-SCTP policy ([RFC3758] refers to them as 'PR-SCTP Services'). [RFC3758] also defines one particular PR-SCTP policy, called Timed Reliability. This allows the sender to specify a timeout for a user message after which the SCTP stack abandons the user message.

This document specifies the following two additional PR-SCTP policies:

Limited Retransmission Policy: Allows to limit the number of retransmissions.

Priority Policy: Allows to discard lower priority messages if space for higher priority messages is needed in the send buffer.

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. Additional PR-SCTP Policies

This section defines two new PR-SCTP policies, one in each subsection.

Please note that it is REQUIRED to implement [RFC3758], if you want to implement these additional policies. However, these additional policies are OPTIONAL when implementing [RFC3758].

3.1. Limited Retransmissions Policy

Using the Limited Retransmission Policy allows the sender of a user message to specify an upper limit for the number of retransmissions for each DATA chunk of the given user messages. The sender MUST abandon a user message if the number of retransmissions of any of the DATA chunks of the user message would exceed the provided limit. The sender MUST perform all other actions required for processing the retransmission event, such as adapting the congestion window and the retransmission timeout. Please note that the number of retransmissions includes both fast and timer-based retransmissions.

The sender MAY limit the number of retransmissions to 0. This will result in abandoning the message when it would get retransmitted for the first time. The use of this setting provides a service similar to UDP, which also does not perform any retransmissions.

Please note that using this policy does not affect the handling of the thresholds 'Association.Max.Retrans' and 'Path.Max.Retrans' as specified in Section 8 of [RFC4960].

The WebRTC protocol stack (see [I-D.ietf-rtcweb-data-channel]), is an example of where the Limited Retransmissions Policy is used.

3.2. Priority Policy

Using the Priority Policy allows the sender of a user message to specify a priority. When storing a user message in the send buffer while there is not enough available space, the SCTP stack at the sender side MAY abandon other user message(s) of the same SCTP

association (with the same or a different stream) with a priority lower than the provided one. User messages sent reliable are considered having a priority higher than all messages sent with the Priority Policy. The algorithm for selecting the message(s) being abandoned is implementation specific.

After lower priority messages have been abandoned high priority messages can be transferred without the send call blocking (if used in blocking mode) or the send call failing (if used in non-blocking mode).

The IPFIX protocol stack (see [RFC7011]) is an example of where the Priority Policy can be used. Template records would be sent with full reliability, while billing, security-related, and other monitoring flow records would be sent using the Priority Policy with varying priority. The priority of security related flow-records would be chosen higher than the the priority of monitoring flow records.

4. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to support the newly defined PR-SCTP policies, to provide some statistical information and to control the negotiation of the PR-SCTP extension during the SCTP association setup.

Please note that this section is informational only.

4.1. Data Types

This section uses data types from [IEEE.1003-1G.1997]: `uintN_t` means an unsigned integer of exactly N bits (e.g. `uint16_t`). This is the same as in [RFC6458].

4.2. Support for Added PR-SCTP Policies

As defined in [RFC6458], the PR-SCTP policy is specified and configured by using the following `sctp_prinfo` structure:

```
struct sctp_prinfo {
    uint16_t pr_policy;
    uint32_t pr_value;
};
```

When the Limited Retransmission Policy described in Section 3.1 is used, `pr_policy` has the value `SCTP_PR_SCTP_RTX` and the number of retransmissions is given in `pr_value`.

When using the Priority Policy described in Section 3.2, `pr_policy` has the value `SCTP_PR_SCTP_PRIO`. The priority is given in `pr_value`. The value of zero is the highest priority and larger numbers in `pr_value` denote lower priorities.

The following table summarizes the possible parameter settings defined in [RFC6458] and this document:

<code>pr_policy</code>	<code>pr_value</code>	Specification
<code>SCTP_PR_SCTP_NONE</code>	Ignored	[RFC6458]
<code>SCTP_PR_SCTP_TTL</code>	Lifetime in ms	[RFC6458]
<code>SCTP_PR_SCTP_RTX</code>	Number of retransmissions	Section 3.1
<code>SCTP_PR_SCTP_PRIO</code>	Priority	Section 3.2

4.3. Socket Option for Getting the Stream Specific PR-SCTP Status (`SCTP_PR_STREAM_STATUS`)

This socket option uses `IPPROTO_SCTP` as its level and `SCTP_PR_STREAM_STATUS` as its name. It can only be used with `getsockopt()`, but not with `setsockopt()`. The socket option value uses the following structure:

```
struct sctp_prstatus {
    sctp_assoc_t sprstat_assoc_id;
    uint16_t sprstat_sid;
    uint16_t sprstat_policy;
    uint64_t sprstat_abandoned_unsent;
    uint64_t sprstat_abandoned_sent;
};
```

`sprstat_assoc_id`: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets this parameter indicates for which association the user wants the information. It is an error to use `SCTP_{CURRENT|ALL|FUTURE}_ASSOC` in `sprstat_assoc_id`.

`sprstat_sid`: This parameter indicates for which outgoing SCTP stream the user wants the information.

`sprstat_policy`: This parameter indicates for which PR-SCTP policy the user wants the information. It is an error to use `SCTP_PR_SCTP_NONE` in `sprstat_policy`. If `SCTP_PR_SCTP_ALL` is used, the counters provided are aggregated over all supported policies.

`sprstat_abandoned_unsent`: The number of user messages which have been abandoned using the policy specified in `sprstat_policy` on the

stream specified in `sprstat_sid` for the association specified by `sprstat_assoc_id`, before any part of the user message could be sent.

`sprstat_abandoned_sent`: The number of user messages which have been abandoned using the policy specified in `sprstat_policy` on the stream specified in `sprstat_sid` for the association specified by `sprstat_assoc_id`, after a part of the user message has been sent.

There are separate counters for unsent and sent user messages because the `SCTP_SEND_FAILED_EVENT` supports a similar differentiation. Please note that an abandoned large user message requiring an SCTP level fragmentation is reported in the `sprstat_abandoned_sent` counter as soon as at least one fragment of it has been sent. Therefore each abandoned user message is either counted in `sprstat_abandoned_unsent` or `sprstat_abandoned_sent`.

If more detailed information about abandoned user messages is required, the subscription to the `SCTP_SEND_FAILED_EVENT` is recommended. Please note that some implementations might choose not to support this option, since it increases the resources needed for an outgoing SCTP stream. For the same reasons, some implementations might only support using `SCTP_PR_SCTP_ALL` in `sprstat_policy`.

`sctp_opt_info()` needs to be extended to support `SCTP_PR_STREAM_STATUS`.

4.4. Socket Option for Getting the Association Specific PR-SCTP Status (`SCTP_PR_ASSOC_STATUS`)

This socket option uses `IPPROTO_SCTP` as its level and `SCTP_PR_ASSOC_STATUS` as its name. It can only be used with `getsockopt()`, but not with `setsockopt()`. The socket option value uses the same structure as described in Section 4.3:

```
struct sctp_prstatus {
    sctp_assoc_t sprstat_assoc_id;
    uint16_t sprstat_sid;
    uint16_t sprstat_policy;
    uint64_t sprstat_abandoned_unsent;
    uint64_t sprstat_abandoned_sent;
};
```

`sprstat_assoc_id`: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets this parameter indicates for which association the user wants the information. It is an error to use `SCTP_{CURRENT|ALL|FUTURE}_ASSOC` in `sprstat_assoc_id`.

`sprstat_sid`: This parameter is ignored.

`sprstat_policy`: This parameter indicates for which PR-SCTP policy the user wants the information. It is an error to use `SCTP_PR_SCTP_NONE` in `sprstat_policy`. If `SCTP_PR_SCTP_ALL` is used, the counters provided are aggregated over all supported policies.

`sprstat_abandoned_unsent`: The number of user messages which have been abandoned using the policy specified in `sprstat_policy` for the association specified by `sprstat_assoc_id`, before any part of the user message could be sent.

`sprstat_abandoned_sent`: The number of user messages which have been abandoned using the policy specified in `sprstat_policy` for the association specified by `sprstat_assoc_id`, after a part of the user message has been sent.

There are separate counters for unsent and sent user messages because the `SCTP_SEND_FAILED_EVENT` supports a similar differentiation. Please note that an abandoned large user message requiring an SCTP level fragmentation is reported in the `sprstat_abandoned_sent` counter as soon as at least one fragment of it has been sent. Therefore each abandoned user message is either counted in `sprstat_abandoned_unsent` or `sprstat_abandoned_sent`.

If more detailed information about abandoned user messages is required, the usage of the option described in Section 4.3 or the subscription to the `SCTP_SEND_FAILED_EVENT` is recommended.

`sctp_opt_info()` needs to be extended to support `SCTP_PR_ASSOC_STATUS`.

4.5. Socket Option for Getting and Setting the PR-SCTP Support (`SCTP_PR_SUPPORTED`)

This socket option allows the enabling or disabling of the negotiation of PR-SCTP support for future associations. For existing associations it allows to query whether PR-SCTP support was negotiated or not on a particular association.

Whether PR-SCTP is enabled or not per default is implementation specific.

This socket option uses `IPPROTO_SCTP` as its level and `SCTP_PR_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 PR-SCTP whereas a value of 0 encodes the disabling of PR-SCTP.

`sctp_opt_info()` needs to be extended to support `Sctp_PR_SUPPORTED`.

5. IANA Considerations

This document requires no actions from IANA.

6. Security Considerations

This document does not add any additional security considerations in addition to the ones given in [RFC4960], [RFC3758], and [RFC6458]. As indicated in the Security Section of [RFC3758], transport layer security in the form of TLS over SCTP (see [RFC3436]) can't be used for PR-SCTP. However, DTLS over SCTP (see [RFC6083]) could be used instead. If DTLS over SCTP as specified in [RFC6083] is used, the security considerations of [RFC6083] do apply. It should also be noted that using PR-SCTP for an SCTP association doesn't allow that association to behave more aggressively than an SCTP association not using PR-SCTP.

7. Acknowledgments

The authors wish to thank Benoit Claise, Spencer Dawkins, Stephen Farrell, Gorrry Fairhurst, Barry Leiba, Karen Egede Nielsen, Ka-Cheong Poon, Dan Romascanu, Irene Ruengeler, Jamal Hadi Salim, Joseph Salowey, Brian Trammell, and Vlad Yasevich for their invaluable comments.

8. References

8.1. Normative References

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

- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

8.2. Informative References

- [RFC3436] Jungmaier, A., Rescorla, E., and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", RFC 3436, December 2002.
- [RFC6083] Tuexen, M., Seggelmann, R., and E. Rescorla, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC 6083, January 2011.
- [RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, December 2011.
- [RFC7011] Claise, B., Trammell, B., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, September 2013.
- [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.
- [IEEE.1003-1G.1997]
Institute of Electrical and Electronics Engineers,
"Protocol Independent Interfaces", IEEE Standard 1003.1G,
March 1997.

Authors' Addresses

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
DE

Email: tuexen@fh-muenster.de

Robin Seggelmann
T-Systems International GmbH
Fasanenweg 5
70771 Leinfelden-Echterdingen
DE

Email: rfc@robin-seggelmann.com

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
US

Email: randall@lakerest.net

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: Salvatore.Loreto@ericsson.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 21, 2016

K. Nielsen
R. De Santis
Ericsson
A. Brunstrom
Karlstad University
M. Tuexen
Muenster Univ. of Appl. Science
R. Stewart
Netflix, Inc.
October 19, 2015

SCTP Tail Loss Recovery Enhancements
draft-nielsen-tsvwg-sctp-tlr-02.txt

Abstract

Loss Recovery by means of T3-Retransmission has significant detrimental impact on the delays experienced through an SCTP association. The throughput achievable over an SCTP association also is negatively impacted by the occurrence of T3-Retransmissions. The present SCTP Fast Recovery algorithms as specified by [RFC4960] are not able to adequately or timely recover losses in certain situations, thus resorting to loss recovery by lengthy T3-Retransmissions or by non-timely activation of Fast Recovery. In this document we specify a number of enhancements to the SCTP Loss Recovery algorithms which amends some of these deficiencies with a particular focus on Loss Recovery for drops in Traffic Tails. The enhancements supplement the existing algorithms of [RFC4960] with proactive probing and timer driven activation of the Fast Retransmission algorithm as well as a number of enhancements of the Fast Retransmission algorithm in itself are specified.

Status of This Memo

This Internet-Draft 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 21, 2016.

Copyright Notice

Copyright (c) 2015 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. The SCTP TLR Function	4
1.1.1. Dependencies	5
1.2. Relation to other work	5
1.2.1. Early Retransmit and RTO Restart	5
1.2.2. TCP applicability	6
1.2.3. Packet Re-ordering	6
1.2.4. Congestion Control	7
1.2.5. CMT-SCTP Applicability	7
2. Conventions and Terminology	8
3. Description of Algorithms	9
3.1. SCTP Scoreboard and miss indication Counting Enhancement	9
3.1.1. Multi-Path Considerations	11
3.2. RFC6675 nextseg() Tail Loss Enhancements for SCTP FR	11
3.2.1. Multi-Path Considerations	14
3.3. SCTP-TLR Description	15
3.3.1. Principles	15
3.3.2. SCTP - TLR Statemachine	19
3.3.3. TLPP Transmission Rules	24
3.3.4. Masking of TLPP Recovered Losses	28
3.3.5. Elimination of unnecessary DELAY-ACK delays	30
4. Confirmation of support for Immediate SACK	31
5. Socket API Considerations	31
6. Security Considerations	31
7. Acknowledgements	32
8. IANA Considerations	32
9. Discussion and Evaluation of function	32
10. References	32
10.1. Normative References	32

10.2. Informative References	33
Appendix A. Unambiguous SACK	35
A.1. TSN Retransmission ID in Data Chunk Header	35
A.1.1. Sender side behaviour	36
A.1.2. Receiver side behaviour	36
A.2. Unambiguous SACK Chunk	36
A.2.1. Receiver side behaviour	40
A.3. Unambiguous SACK return	40
A.4. Negotiation	41
Authors' Addresses	41

1. Introduction

Loss Recovery by means of T3-Retransmission has significant impact on the delays experienced through, as well as, the throughput achievable over an SCTP association. Loss Recovery by Fast Retransmission operation in many situations is superior to T3-Retransmission from both a latency and a throughput perspective.

The present SCTP Fast Retransmission algorithm, as specified by [RFC4960], is driven uniquely by exceed of a DupTresh number of miss indication counts stemming for returned SACKs, and it is as such not able to adequately or timely recover losses in traffic tails where a sufficient number of such SACKs may not be generated, there resorting to loss recovery by T3-Retransmissions or by non-timely activation of Fast Recovery. Non-timely activation here refer to the situation where activation of Fast Recovery for packets lost within one data burst needs to await arrival of SACKs from a subsequent data burst.

By drop in traffic tails (or tail drops) we refer generally and specifically to the following situations:

1. Drops of the last SCTP packets of an SCTP association or more generally drop of packets in the end of an SCTP association which are not proceeded by more than DupThresh number of packets which are not dropped.
2. Drops among packets sent in a the end of bursts spaced by pauses of time equal to or greater than the T3-timeout (approximately). It is noted that such bursts (pauses in between bursts) may result from application limitations, from congestion control limitations or from receiver side limitations.
3. Drops among packets sent so sparsely that each dropped packet constitutes a tail drop in that DupThresh number of packets would not be sent (would not be available for sent) prior to expiry of the T3-timeout.

It shall be noted that while the above traffic drop criteria describe drops among the forward data packets only, then drops among forward data packets combined with drops of the returned SACKs may together result in that an insufficient number of SACKs be returned to traffic sender for that the Fast Retransmission algorithm be activated prior to T3-timeout occurring. The tail traffic situations for which SCTP Fast Retransmission is not able to recover the losses is thus in general broader than the exact situations listed above. The improvements specified include enhancement of SCTP to deduce the miss indication counts from enhanced scoreboard information thus removing some of the vulnerability of the present SCTP miss indication counting to loss of SACKs.

1.1. The SCTP TLR Function

The function proposed for enhancements of the SCTP Loss Recovery operation for Traffic Tail Losses is divided in two parts:

- o Enhancements of SCTP Fast Retransmission (SCTP FR) algorithm by means of the following Tail Loss Recovery improving functions inspired by or specified by [RFC6675] for TCP:
 - * miss indication counting for a missing (non-SACK'ed) TSN will be based on augmented scoreboard information such that the miss indications will be based not on the number of returned SACKs but on the number of SACK'ed SCTP packets carrying data chunks of higher TSNs. The mechanism is specified both in terms of packets, the book-keeping of which requires new logic, as well as in terms of a less implementation demanding byte based variant following the Islost() approach of [RFC6675]. We shall refer to this improvement as Extended miss indication Counting.
 - * Fast Recovery operation is extended to include the "last resort" retransmission, Nextseg 3) and Nextseg 4), operations of [RFC6675], thus supporting conditional proactive fast retransmissions of missing, but not yet classified as lost, TSNs within the Fast Recovery Exit Point.
- o New SCTP Tail Loss Recovery State machine with proactive timer driven activation of (the enhanced) Fast Recovery operation. Timer driven activation of Fast Recovery is initiated for outstanding data whenever a certain time, shorter than the T3 timeout, has elapsed from the transmittal of the lowest outstanding TSN and network responsiveness, in form of SACKs of packets ahead of the TSN, has been proven since the transmittal of the lowest outstanding TSN. The SCTP TLR mechanism implements a new timer, the Tail Loss Probe timer (PTO), and it works in parts by:

- * Forced activation of Fast Recovery when network responsiveness has been proven, and the PTO timer has kicked, since transmittal of the lowest outstanding TSN, but additional traffic sent (SACKs of TSNs ahead of the TSN) has not served to activate Fast Recovery based on the Extended Mis Indication Counting.
- * Probing for network responsiveness, by transmittal of a TLR probe packet, when no network responsiveness information (no SACKs have been received for any packets ahead of line of the TSN) is available at expiration of the PTO timer relative to the lowest outstanding TSN
- * Activation for T3-retransmission Loss Recovery only when the network remains unresponsive (no SACKs are received) also after transmittal, and subsequently timeout, of a TLR probe packet.

1.1.1. Dependencies

The SCTP TLR procedures proposed apply as add-on supplements to any SCTP implementation based on [RFC4960]. The SCTP TLR procedures in their core are sender-side only and do not impact the SCTP receiver.

Exploitation of SCTP immediate SACK feature, [RFC7053], and usage of new (to be defined) Unambiguous Selective Acknowledgement feature of SCTP require support in both sender and receiver of these SCTP extensions.

1.2. Relation to other work

1.2.1. Early Retransmit and RTO Restart

It is noted that the Early Retransmit algorithm, [RFC5827], addresses activation of Fast Recovery for a particular subset of the tail drop situations in target of the SCTP TLR function. The solution proposed embeds (as a special case) the Early Retransmits algorithm in the delayed variant, experienced with for TCP in [DUKKIPATI02] in which Early Retransmission is only activated provided a certain time has elapsed since the lowest outstanding TSN was transmitted. The delay adds robustness towards spurious retransmissions caused by "mild" packet re-ordering as documented for TCP in [DUKKIPATI02].

It is further noted that depending on the exact situation (e.g., drop pattern, congestion window and amount of data in flight) then T3-retransmission procedures need not be inferior to Fast Retransmission procedures. Rather in some situations T3-retransmission will indeed be superior as T3-retransmissions allow for ramp up of the congestion window during the recovery process.

The changes proposed in this document focus on improving the Loss Recovery operation of SCTP by enforcing timely activation of (improved) Fast Retransmission algorithms. With the purpose to reduce the latency of the TCP and SCTP Loss Recovery operation [HURTIG] has taken the alternative approach of accelerating the activation of T3-retransmission processes when Fast Recovery is not able to kick in to recover the loss. [HURTIG] only addresses a subset of the Tail loss scenarios in scope in the work presented here. The ideas of [HURTIG] for accurate RTO restart are drawn on in the solution proposed here for accurate restart of the new tail loss probe timer (PTO-timer) as well as for accurate set of the T3-timer under certain conditions thus harvesting some of the same latency optimizations as [HURTIG]. The same approach has recently been exploited for TCP by the invention of the TLPR function by the authors of [Rajiullah].

1.2.2. TCP applicability

SCTP Loss Recovery operation in its core is based on the design of Loss Recovery for TCP with SACK enabled. The enhancements of SCTP Tail Loss Recovery proposed here are applicable for TCP.

Note: The - to be determined - exploitation of SCTP immediate SACK feature, [RFC7053], and the - to be determined - usage of new unambiguous selective acknowledgement feature of SCTP may not be readably applicable to TCP at present. ISSUE: Need to follow up on [zimmermann02], [zimmermann03],

It is noted that while the SCTP TLR algorithms and SCTP TLR state machine defined is inspired by the timer driven tail loss probe approach specified in [DUKKIPATI01] for TCP, then the solution defined here differs in the approach taken. The approach here is a clean state approach defining a new comprehensive SCTP TLR state machine as an add-on to the (at least conceptually) existing Fast Recovery and T3-Retransmission SCTP state machines of SCTP. Thereby the SCTP TLR algorithm is able to address all tail loss patterns, whereas the approach of [DUKKIPATI01] relies on a number of experimental mechanisms ([DUKKIPATI02], [MATHIS], [RFC5827]) defined for TCP in IETF or in Research with ad hoc extension to support selected tail loss patterns by addition of the tail loss probe mechanism and the therefrom driven activation of the mechanisms.

1.2.3. Packet Re-ordering

The solution proposed is an enhancement of the existing mis indication counting based Fast Recovery operation of SCTP, [RFC4960], and as such the solution inherits the fundamental vulnerability to

packet re-ordering that the SCTP Fast Retransmission algorithm of [RFC4960] embeds.

For deployment of SCTP in environments where the Fast Retransmission algorithm of [RFC4960] gives rise to spurious entering of Fast Recovery it would be relevant to look into remedies which may detect such and undo the effects of such. Possibly following the approaches taken for TCP (and SCTP) in this area.

OPEN ISSUE: In severe packet re-ordering situations where the second packet of two subsequently sent packets outrace the first packet in arrival with more than PTO time, then such may trigger the SCTP TLR function to enter spurious Fast Recovery. It is conjectured that the this situation does not significantly increase the vulnerability of Loss Recovery to packet-reordering. To be determined and evaluated.

1.2.4. Congestion Control

In its very nature of prompting for activation of Fast Recovery instead of T3-Retransmission Recovery then the benefit of the solution proposed versus the existing solution of [RFC4960] will depend on the CC operation not only during the recovery process but also after exit of the recovery process. In this context it is noted that the prior approach taken for TCP, [DUKKIPATI01], has been documented for a TCP implementation running CUBIC, e.g., see [zimmermann01], whereas SCTP runs a CC algorithm more similar to TCP Reno CC as defined by [RFC5681].

The solution at present is defined within the constraints of existing Congestion Control principles of STCP as defined by [RFC4960]. It is anticipated that Congestion Control improvements are desirable for SCTP in general as well as for the functions defined here in particular.

1.2.5. CMT-SCTP Applicability

The SCTP TLR specification in this document applies to a SCTP implementation following the [RFC4960] principles of using one shared SACK clock spanning the data transfer over multiple paths. It is noted that in its nature of maintaining the common SACK clock principles of [RFC4960] then the SCTP TLR mechanism specified here retains some of the vulnerabilities from [RFC4960] to spurious (or delayed) entering of Fast Recovery operation caused by path changes in inhomogeneous environments (change of data transfer among paths of significantly different RTTs). The validity of this choice is motivated by that concurrent data transfer on multiple paths is the exception case in [RFC4960] MH SCTP and remains the exception also with the enhancements of [RFC4960] specified here.

It is envisaged that the SCTP TLR mechanism specified is readably applicable also to a SCTP implementation supporting concurrent multi path transfer in line with the specification of [CMT-SCTP]. Though is it emphasized that SCTP-TLR, when applied to [CMT-SCTP], needs some adjustments as it should be applied in a split manner following the principles of SFR of [CMT-SCTP].

2. Conventions and Terminology

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

For the purposes of defining the SCTP TLR function, we use the following terms and concepts:

"DupThresh": The number of miss indication counts on an outstanding TSN at the reach of which SCTP declares the TSN as lost and enters Fast Recovery for the TSN if not in Fast Recovery already.

"Flight size": At any given time we define the "Flight size" to be the number of bytes that a SCTP sender considers to be in flight in the network from the sender to the receiver. It is noted that the bytes of a message, which is considered lost and which has not been retransmitted, is not contained in the Flight size. Further it is noted that the bytes of a message which has been retransmitted (once) will count either once or twice in the Flight size depending on whether SCTP considers the first transmission of the message as having been lost (dropped) in the network.

"Outstanding TSN": A TSN (and the associated DATA chunk) that has been sent by the SCTP sender for which it has not yet received an acknowledgement and which the SCTP sender has not abandoned (e.g., abandoned as a result of [RFC3758]).

"highTSN": The highest outstanding TSN at this point in time.

"lowTSN": The lowest outstanding TSN at this point in time.

"Scoreboard": An SCTP sender need maintain a data structure to store various information on a per outstanding TSN basis. This includes the selective acknowledgment information, miss indication counts, bytes counts and other information defined [RFC4960], in this document and in other SCTP specifications. This data structure we refer to as "scoreboard". The specifics of the scoreboard data structure are out of scope for this document (as

long as the implementation can perform all functions required by this specification).

3. Description of Algorithms

3.1. SCTP Scoreboard and miss indication Counting Enhancement

Entering of Fast Recovery in SCTP, as specified by [RFC4960]), is driven by miss indication counts. When a TSN has received DupThresh=3 miss indication counts, the TSN is declared lost and will be eligible for fast retransmission via Fast Recovery procedure.

miss indication counts are in RFC4960 SCTP driven entirely by receipt of SACKs in accordance with the Highest TSN Newly Acknowledged algorithm (section 7.2.4 of [RFC4960]):

Highest TSN Newly Acknowledged (HTNA): For each incoming SACK, miss indications are incremented only for missing TSNs prior to the highest TSN newly acknowledged in the SACK. A newly acknowledged DATA chunk is one not previously acknowledged in a SACK.

An evident issue with the HTNA algorithm is that it is vulnerable to loss of SACKs. In many situations loss of SACKs will result only in a slight delayed entering of Fast Recovery for a dropped TSN, but generally, then by relying on HTNA algorithm only, loss of SACKs will further broaden the traffic tails situations where Fast Recovery either not be activated in a timely manner or not be activated at all due to the receipt of an insufficient number SACKs only.

In order to make SCTP Fast Recovery more robust towards drop of SACKs, the following extension of the HTNA algorithm SHOULD be supported by an SCTP implementation:

Newly Acked Packets ahead-of-line (NAPahol): For each incoming SACK, miss indications are incremented only for missing TSNs prior to the highest TSN newly acknowledged in the SACK. A newly acknowledged DATA chunk is one not previously acknowledged in a SACK. For each missing TSN thus potentially eligible for additional miss indication counts, the number of miss indications to be given shall follow the number of newly acknowledged packets ahead of line of the packet of the missing TSN.

The solution is robust towards split SACK. The solution requires for the SCTP implementation to keep track of the relationship in between data chunks (TSN numbers) and packets. One solution is for the SCTP implementation to maintain a packet id as a monotonically incrementing packet sequence number to map chunks to packets and for

each outstanding chunk to keep state of the packet id that the chunk was sent in as well as (incrementally updated) the packet ids of up to DupThresh-1 (=2) packets ahead of line for which chunks have been SACKed.

For accurate PTO-timer management, using the restart principles of [HURTIG] and [Rajiullah], see Section 3.3, an SCTP TLR implementation is required to keep track of the time at which packets/TSNs are transmitted (or strictly speaking to be able to deduce the time since a packet/a TSN was last transmitted). An implementation may exploit timestamps for the generation of (part of) the packet id as well as for the mentioned time management thereby limiting the additional overhead required for the packet id storage.

As an alternative to the above accurate packet counting then an SCTP implementation MAY, to reduce implementation complexity, instead support the following bytes counting based extension of the RFC4960 HTNA algorithm:

Highest Bytes Newly Acknowledged (HBNA): For each incoming SACK, miss indications are incremented only for missing TSNs prior to the highest TSN newly acknowledged in the SACK. A newly acknowledged DATA chunk is one not previously acknowledged in a SACK. For each missing TSN thus eligible for additional miss indication counts, the number of miss indications to be given shall follow the number of newly acknowledged bytes in the SACK ahead of line of the missing TSN in the following manner Add-miss indication-count(TSN) = Ceiling((Newly bytes ahead of line(TSN))/PMTU).

The HBNA approach as specified above is vulnerable to split of SACK. An implementation choice which is robust to split of SACK is to recalculate the total amount of selectively acknowledged bytes ahead of line of an outstanding TSN and update the miss indication count of the TSN as Ceiling((Selectively Aced bytes ahead of line (TSN))/PMTU). This more robust implementation choice however demands either for maintain of additional state per TSN, namely the Selectively Aced bytes ahead of line (TSN) or for extensive repeated computations. Risk of split SACK may not be weighty enough to worth such implementation complexity.

The HBNA approach follows the approach taken for TCP, Islost(), in [RFC6675]. It is noted, however, that due to the message based approach of SCTP, then a byte based approach generally will be less accurate as a measure for the number of packet received ahead of line than it is for byte stream based TCP.

3.1.1. Multi-Path Considerations

In multi-homed [RFC4960] SCTP, data that potentially will be subject to fast retransmission may be in flight on multiple paths. This (exception) situation can occur as a result of a change of the data transfer path, which may come about, e.g., as a result of a switchback operation performed autonomously by SCTP or as a result of a management operation setting a new primary path. The situation can also occur as a result of destination directed data transfer where the destination address specified is different from the present data transfer path destination. In an [RFC4960] SCTP implementation, SACKs of data sent on one path will increase the miss indication counts of data with lower TSN in flight on a different path. As such SACKs of data sent on one path may actually result in generation of (potentially spurious) loss event reactions on a different path. This fundamental aspect of [RFC4960] miss indication counting is not changed in this document. Meaning that it is not intended for the miss indication counting improvements defined above, i.e., the NAPahol and the HBNA mechanisms, to discriminate among the paths on which the SACK'ed data contributing to the miss indication counting has been sent.

3.2. RFC6675 nextseg() Tail Loss Enhancements for SCTP FR

The Fast Retransmission algorithm for TCP as specified in [RFC6675] implements some differences compared to the Fast Retransmission algorithm specified for SCTP by [RFC4960]. Of particular significance for recovery of losses in traffic tail scenarios are the fact that the [RFC6675] algorithm, once Fast Recovery has been activated, takes two "last resort" retransmission measures, step 3) and step 4) of Nextseg() of [RFC6675]. These measures facilitate the recovery of losses in situations where only an insufficient number of SACKs would be able to be generated to complete the Fast Recovery process without resorting to T3-timeout. For SCTP Fast Recovery we formulate the equivalent measures as follows:

Last Resort Retransmission: If the following conditions are met:

- * there are no outstanding TSN's eligible for fast retransmission due to DupThresh or more miss indications
- * there is no new data available for transmission

then an outstanding TSN less than or equal to the Fast Recovery Exit Point, for which there exists SACKs of chunks ahead of line of the TSN, may be retransmitted provided the CWND allow. The bytes of a TSN which is retransmitted in this manner are not subtracted from the Flight size prior to this action be taken nor

as a result of this action. If the miss indication count of the TSN subsequently reaches the DupThresh value, the bytes of the TSN shall be subtracted from the Flight size. Once acknowledged the remaining contribution of this TSN in the Flight size (whether it be there counted once or twice at this point in time) is subtracted. A TSN which is retransmitted in this manner will be marked as ineligible for a subsequent fast retransmit (see considerations on Multiple Fast Retransmission operation in Section 3.3.1.3).

An SCTP implementation which implements the Unambiguous SACK feature of Appendix A may implement a more accurate calculation of the flightsize when doing Last Resort Retransmission. That is, instead of subtracting the contribution from the retransmitted TSN from the flightsize once the acknowledgement of the TSN arrives, the SCTP implement may distinguish where the acknowledgment is for the original TSN or for the retransmitted TSN and in case the acknowledgement is not for the retransmitted TSN, SCTP should delay the subtract of the bytes of the retransmitted TSN from the flightsize until either an acknowledgement of the retransmitted TSN is received (see Appendix A) or until PTO2-T_latest(TSN) time has elapsed (see Section 3.3.1).

Rescue: If all of the following conditions are met:

- * there are no outstanding TSN's eligible for fast retransmission due to DupThresh or more miss indications
- * there is no new data available for transmission and no data is outstanding on the association beyond the Fast Recovery Exit Point
- * there are no outstanding TSNs eligible for Last Resort Retransmission
- * the cumack has progressed since this entering of Fast Recovery

and there exist non-SACKed, non fast retransmitted TSNs, within the Fast Recovery Exit point, then for this entry of Fast Recovery, conditionally to that the CWND allows, we allow for fast retransmission of one packet of consecutive outstanding non fast retransmitted TSNs up to PMTU size, the highest TSN of which MUST be the highest outstanding TSN within the Fast Recovery Point. The bytes of a TSN which is retransmitted in this manner are not subtracted from the Flight size prior to this action be taken nor as a result of this action. If the miss indication count of the TSN subsequently reaches the DupThresh value, the bytes of the TSN shall be subtracted from the Flight size. Once acknowledged the

remaining contribution of this TSN in the Flight size (whether it be there counted once or twice at this point in time) is subtracted. A TSN which is retransmitted in this manner will be marked as ineligible for a subsequent fast retransmit (see considerations on Multiple Fast Retransmission operation in Section 3.3.1.3).

An implementation of the Rescue operation may be accomplished by maintain of an RescueRTX parameter as described for TCP in [RFC6675].

An SCTP implementation which implements the Unambiguous SACK feature of Appendix A may implement a more accurate calculation of the flightsize when performing Rescue operation. That is, instead of subtracting the contribution from the retransmitted TSN from the flightsize once the acknowledgement of the TSN arrives, the SCTP implement may distinguish where the acknowledgment is for the original TSN or for the retransmitted TSN and in case the acknowledgement is not for the retransmitted TSN, SCTP should delay the subtract of the bytes of the retransmitted TSN from the flightsize until either an acknowledgement of the retransmitted TSN is received (see Appendix A) or until $PTO2 - T_{latest}(TSN)$ time has elapsed (see Section 3.3.1).

DISCUSSION: [RFC4960] in addition to the HTNA algorithm demand for additional miss indication counting to be performed during Fast Recovery according to the following prescription (section 7.2.4 of [RFC4960]):

- (#) If an endpoint is in Fast Recovery and a SACK arrives that advances the Cumulative TSN Ack Point, the miss indications are incremented for all TSNs reported missing in the SACK.

It is noted that under special circumstances then (#) makes SCTP Fast Recovery complete in situations where TCP Fast Recovery would only complete by virtue of the measure 3) or 4) of [RFC6675] and as such these measures are more critically demanded for TCP Fast Recovery operation than for the SCTP Fast Recovery operation. However as documented by (OPEN ISSUE: to be filled in) the Last Resort Retransmission operation and the Rescue operation also for SCTP significantly improve the Loss Recovery operation; the latency of the individual loss recovery operation as well as the ability of the operation to complete without resort to T3-timeout. Consequently this document prescribes for SCTP TLR to implement these procedures. Conversely even when the measures 3) and 4) of [RFC6675] are implemented, (#) gives benefits in terms of releasing flight size space allowing Fast Recovery to progress.

As the algorithm extension is limited by the existing congestion control algorithm of SCTP, these extensions of SCTP Fast Recovery do not compromise the TCP fairness of the SCTP Fast Recovery Operation.

3.2.1. Multi-Path Considerations

In multi-homed [RFC4960] SCTP, data that potentially will be subject to Fast Retransmission may be in flight on multiple paths. This (exception) situation in particular can occur as a result of a change of the data transfer path as a result of a switchback operation to a primary path. Here SACKs of data sent on one path (e.g., the new data transfer path) may result in generation of (potentially spurious) loss event reactions on a different path (the prior data transfer path). The [RFC4960] miss indication counting based on a common SACK clock is not changed in this document, nevertheless the protocol operation, here the operation of the Last Resort Retransmission and the Rescue operation in this situation, need to be specified.

The specification in this document is based on the following fundamental goals:

- o an [RFC4960] SCTP implementation must appropriately react to loss events observed by means of miss indication counting, by performing appropriate adjustments of CWND and sstresh, on all paths where such loss events are observed.
- o The observation of a loss event on one path should not for [RFC4960] SCTP MH impact the congestion control operation on a different path.

For the implementation of the Last Resort Retransmission and the Rescue operations for [RFC4960] MH SCTP then the following specifications are given:

- o For a TSN to be eligible for Last Resort Retransmission a loss event MUST have been observed on the path on which this TSN is in flight.
- o For a TSN to be eligible for the Rescue operation a loss event MUST have been observed on the path on which this TSN is in flight.

An implementation of the above may be accomplished by the implementation of a Fast Recovery state and Fast Recovery Exit point on a per path basis with the following particulars:

- o A path enters the Fast Recovery State based on loss event observation of TSNs in flight on the path.
- o When a loss event is observed on a path the Fast Recovery Exit point on the path is set to the highest TSN in flight of the path.
- o Fast Retransmission of TSNs in flight on the path terminates once the Fast Recovery Exit Point on the path has been reached (i.e., has been cumulative SACK'ed) at which point the Fast Recovery process on the path is terminated.
- o The eligibility of a TSN for the Last Resort Retransmission and the Rescue operation shall follow the prescriptions given above with adherence to the Fast Recovery Exit point set on the path on which the TSN is in flight.

The data retransmission process of data chunks in itself is prescribed to happen on the present data transfer path of the association regardless of which path the data chunks were in flight on when they became eligible for Fast Retransmission. This follows [RFC4960] and the preceding [CAR002].

With the above per path modelling of the Fast Recovery operation, SCTP may have multiple fast recovery exit points at any given time (though at most one per path) and the fast recovery operation may terminate at different times on the different paths. Further it is noted that a path may be in Fast Recovery even if no data is in flight on the path or even if the only data in flight on the path is beyond the Fast Recovery Exit Point of the path. The latter can occur in the very peculiar case where fast retransmission of data declared lost on the path happens on a different path as well as that the user performs a data directed data transfer on the path in question.

An SCTP implementation fulfilling the goals described above may also be achieved by other means than by maintain of a per path Fast Recovery Exit point. For example it might be achieved by maintain of a common association Fast Recovery Point spanning multiple paths, but still the implementation must ensure appropriate per destination address congestion control operation.

3.3. SCTP-TLR Description

3.3.1. Principles

The SCTP TLR function is based on the following principles.

3.3.1.1. Retransmission Timers Management

This document is specified as if there is a single retransmission timer per destination transport address, but implementations MAY have a retransmission timer for each DATA chunk.

This document specifies usage of new PTO timer for SCTP TLR. The document is specified as if the PTO timer functions are implemented by means of the existing retransmission timer of [RFC4960] SCTP, i.e., under certain conditions the retransmission-timer is activated with special PTO values rather than with the standard T3-timer value. The document is specified as if there is a single PTO timer per destination transport address, equivalently a single PTO timer per path. Implementations MAY choose to implement a PTO timer per DATA chunk.

For an outstanding TSN we define the time $T_{latest}(TSN)$ to be the time that has elapsed since the TSN was last sent. When a TSN is first sent, or when it is retransmitted, $T_{latest}(TSN)=0$. An SCTP TLR implementation must be able to deduce this value for any outstanding TSN.

3.3.1.2. Timer driven entering of Fast Recovery

Timer driven entering of Fast Recovery in SCTP TLR is based on the following principles:

- o Maintain of a Tail Loss Probe Timer (PTO) which in certain situations (generally when retransmission is not performed) is running on a path. At any given time the value of the PTO timer is related to the lowest TSN in flight on the path. The PTO timer value used will depend on the situation:

By default the following timer value is used:

PTO1: $PTO = \min(RTO, 1.5 * SRTT + \max(RTTVAR, DELAY_ACK))$

Whereas the following value is used:

PTO2: $PTO = \min(RTO, 1.5 * SRTT + RTTVAR)$

when it is known that subsequent SACKs not acknowledging the TSN for which the PTO is running will be (or will have been) returned immediately. For more details see Section 3.3.2.

By design the probe timer is kept lower or equal to the RTO, thereby aiming to prevent a potential unnecessary and damaging RTO, as well as generally larger than an anticipated RTT

thereby preventing that it kicks in prematurely. I.e., the timer only kicks in at a time where one would have expected to have received a SACK of the lowest TSN in flight were there no problems.

A minimal PTO value, PTO_MIN, is applied to the above formulas (particularly important for PTO2). I.e., the effective PTO1 = MAX(PTO_MIN, PTO1) and the effective PTO2 = MAX(PTO_MIN, PTO2). The suggested value of PTO_MIN is 10 msec. In the following when referring to PTO1 and PTO2 we refer to the effective PTO1 and PTO2 values.

For an SCTP implementation which performs RTT measurements during the association set-up, the PTO set on the path on which the first data chunk is sent shall be initialized from the RTT measured on the path during the association set-up. If no such RTT measurement is performed or is available on the particular path in question, the PTO shall be initialized as RTO_INIT.

- o PTO timer driven transmittal of Tail Loss Probe Packet: Once data is outstanding on a path and the PTO timer of the path kicks and no SACKs of any chunks with higher TSN number have arrived, a probe packet, denoted a Tail Loss Probe Packet (TLPP), is sent to probe for network responsiveness (i.e., for SACK of the TLPP) in order to potentially drive proactive entering of Fast Recovery.
- * For a SCTP sender that supports the Immediate SACK feature, [RFC7053], the I-bit MUST be set on chunks sent in a TLPP packet.
- o PTO timer driven entering of Fast Recovery: Process is enforced when network responsiveness is proven (SACK of later sent data than lowest TSN in flight on the path is available) and (at least) PTO time has elapsed since transmittal of this lowest TSN in flight on the path.

Comment: The lowest outstanding TSN on an association may under special circumstances not be in flight on any path of the association. This can happen when the lowest outstanding TSN has been declared lost but the transmittal of the TSN is prevented due to congestion window limitations (e.g., during Fast Recovery). In this case, as well as generally for TSNs that are being retransmitted due to fast retransmission or T3-timeout, no PTO timer is running on the TSN. Conversely when the lowest outstanding TSN on a path is not subject to Fast Recovery or T3-Recovery, then this lowest outstanding TSN is also in flight on the path.

3.3.1.3. Fast-Recovery and Loss Detection

Fast Recovery and miss indication counting for the SCTP TLR function MUST embed the enhancements described in Section 3.2. In addition SCTP TLR implements the following loss detection during Fast Recovery:

- o If in Fast Recovery, then an outstanding TSN in flight on the path, with TSN lower than the Fast Recovery Exit Point on the path, is declared lost when the following conditions are satisfied:
 - * The TSN has not been fast retransmitted.
 - * $T_{latest}(TSN) > PTO2$.
 - * The TSN is lower than the highest outstanding SACK'ed TSN.

When declared lost by this procedure the TSN is subtracted from the flight size as well as it becomes eligible for fast retransmission as if it had been declared lost by reach of Dupthresh miss indication counts.

Such loss detection during SCTP TLR Fast Recovery shall at a minimum be done at receipt of SACK as well as at times where the possibility to transmit new data is being evaluated. An implementation maintaining PTO timers on a per data chunk basis may make further evaluation based on timer expiration.

Following [RFC4960] it is assumed that a data chunk should only be fast retransmitted once. I.e., subsequent retransmissions of the data chunk must proceed as T3-retransmission. An SCTP TLR implementation MAY possibly implement Multiple Fast Retransmission operation following the principles described in [CAR001] extended to include the Last Resort Retransmission and Rescue operations. Such however is not covered by the specification given here.

3.3.1.4. T3-Recovery

[RFC4960] does not explicitly specify for an T3-Recovery phase to be supported for SCTP, nor does [RFC4960] explicitly demand for that a data chunk which has been T3-retransmitted cannot undergo fast retransmission. It can be an advantage that a lost T3-retransmitted data chunk may be recovered by timely fast retransmission rather than by a subsequently, potentially back-off'ed T3-retransmission. For [RFC4960] MH SCTP, however, reliable implementation of such fast recovery of lost T3-retransmitted data is difficult to achieve given the usage of one common SACK clock as new data on one path may trick

spurious fast retransmission of data that has been/is being T3-retransmitted on a different path. Here it is important to emphasize that concurrent T3-retransmission and new data transmission on different paths is the standard operation of MH SCTP [RFC4960]. (Though implementations might possibly mitigate such effects by only sending new data after completion of the T3-retransmission operation as well as the implementation of SCTP-PF, [SCTP-PF], would further decrease the likelihood of such concurrent data transfer occurring.)

In this document we assume that an SCTP implementation follows either of the following implementation choices:

- o A data chunk which has underwent T3-retransmission cannot subsequently be subject to Fast Retransmission whether such entering of Fast Recovery be driven alone by miss indication counting or by the SCTP TLR mechanism. This implementation choice corresponds to implementing a T3-Recovery phase for SCTP equivalent with the RTO-recovery phase of TCP.
- o A data chunk, which has underwent T3-retransmission, will be eligible for subsequent Fast Retransmission if such is driven by miss indication counts from SACKs of new data chunks sent after all data outstanding for T3-retransmission have been sent and the new data is sent on the same path as the T3-retransmission data.

One implementation choice may be to follow the first implementation choice for SCTP MH and the second implementation choice for SCTP SH. Regardless of this implementation choice then in SCTP TLR a data chunk that has been subject to T3-retransmission SHOULD NOT be subject to the timer driven entering of Fast Recovery specified below. The motivation for this choice is that the SRTT may not be appropriately refreshed during the T3-retransmission process. OPEN ISSUE/TO DO: Ideally the PTO timer used after the exit of the T3-recovery phase should be updated based on a fresh RTT measurement. E.g., from the last acknowledged TSN. If no new SRTT calculation is made based on a scheduled RTT measurement, then the PTO timer values could be made sure to be appropriately adjusted, if necessary, by a last measured RTT by $1,5 \cdot \text{SRTT} + \text{RTTVAR} \rightarrow \text{MAX}(1 \cdot 5 \cdot \text{RTT}, 1,5 \cdot \text{SRTT} + \text{RTTVAR})$.

3.3.2. SCTP - TLR Statemachine

The SCTP Tail Loss Recovery function defines 3 states: The SCTP TLR OPEN state, the SCTP TLR PROBE WAIT state and the SCTP TLR DELAY WAIT state. At any given time the SCTP transmission logic for the lowest outstanding TSN on a path will be in one of these 3 states or the TSN is sought being recovered by means of Fast Recovery or T3-Recovery.

Figure 1 illustrates the states and the state transitions.

(to be inserted)

Figure 1, Enhanced Loss Recovery State Machine Diagram

In the following we describe the states and the actions taken.

3.3.2.1. SCTP TLR OPEN STATE

This is the state the SCTP transmission logic is in on any path when no TSN is outstanding on the association as well as it is the state when SCTP sends the first data on a path after idle/no TSN outstanding. It also more generally is the state the transmission logic is in when there are no gaps in the SACK scoreboard beyond the lowest outstanding TSN on the path.

In this state SCTP is not performing Fast Recovery nor T3-Recovery on the lowest TSN outstanding on the path and no SACKs of any chunks with higher TSN number have arrived. In this state, when SCTP has outstanding data on the path, a PTO timer is running relative to the lowest TSN outstanding on the path.

The PTO set on a (new) lowest outstanding TSN on the path in this state will follow PTO1 when less than 2 packets are outstanding beyond the TSN at the time when the timer is set and follow PTO2 when 2 or more packets are outstanding beyond the TSN when the PTO timer is set or when the Immediate SACK feature is known to be supported by both sender and receiver (see Section 4) and the I-bit has been set on the TSN or on an outstanding TSN of higher number.

In the OPEN state the following may happen:

- o A SACK commutatively acknowledging the lowest outstanding TSN and resulting in no gaps in the SACK scoreboard may arrive. In this case the state remains in OPEN state. If there still is outstanding data on the path, the PTO timer is set on the new lowest outstanding TSN. The PTO timer value set will be the value $PTO - T_latest(TSN)$ where the PTO value is calculated either from PTO1 or PTO2 according to the evaluation criteria given above.
- o A SACK with gap(s) may arrive, thus proving network responsiveness while still not cumulatively acknowledging all lower (than the SACK'ed gap) outstanding TSNs on the path. The SACK may or may not move the cumulative ACK point. This indicates that either

packets are being re-ordered or the (new) lowest outstanding TSN on the path has been lost.

- * If the SACK makes the miss indication count on the (new) lowest outstanding TSN reach Dupthresh the SCTP OPEN state is terminated and Fast Recovery is started.
- * If Dupthresh miss indication count is not reached on the (new) lowest outstanding TSN, the state will now transit to SCTP TLR DELAY WAIT state for potential entering of SCTP TLR driven Fast Recovery if the PTO timer kicks prior to the (new) lowest outstanding TSN has been acknowledged or for potential later entering of Fast Recovery by reach of Dupthresh miss indication counts. When transiting to SCTP TLR DELAY WAIT the PTO timer relative to the (new) lowest outstanding TSN is reset to $PTO2 - T_latest(TSN)$. In case $PTO2 - T_latest(TSN) \leq 0$, the DELAY WAIT state is immediately terminated, the packet containing the lowest outstanding TSN is declared lost, and Fast Recovery is started.
- o The PTO timer relative to the lowest outstanding TSN may kick, in which case SCTP TLR will send a TLPP, reset the PTO timer relative to the lowest outstanding TSN to a T3 timer and transit to SCTP TLR PROBE WAIT state to await either the kick of the T3 relative to the lowest outstanding TSN (network is persistently unresponsive) or proof of network responsiveness and potential entering of SCTP TLR driven Fast Recovery unless the network responsiveness proof comes in form of cumulative acknowledgement of the TSN. The T3-value set relative to the lowest outstanding TSN when sending the TLPP probe and entering this state shall be:
 - * $MAX(PTO1, RTO - T_latest(TSN))$, when receiver side support for Immediate SACK has not been confirmed for the association, see Section 4.
 - * $MAX(PTO2, RTO - T_latest(TSN))$, when receiver side support for Immediate SACK has been confirmed for the association, see Section 4, and the SCTP sender itself deploys the Immediate SACK feature.

For further details on the TLPP transmission see Section 3.3.3.

3.3.2.2. SCTP TLR PROBE WAIT STATE

In this state the lowest outstanding TSN has remained unSACK'ed for more than PTO time and no indication (no SACK of higher outstanding TSNS have been received) thus resulting in the transmittal of a TLPP to probe for the network responsiveness.

The T3-value set relative to the lowest outstanding TSN when sending the TLPP probe and entering this state is:

- o $\text{MAX}(\text{PTO1}, \text{RTO} - \text{T_latest}(\text{TSN}))$, when receiver side support for Immediate SACK has not been confirmed for the association, see Section 4.
- o $\text{MAX}(\text{PTO2}, \text{RTO} - \text{T_latest}(\text{TSN}))$, when receiver side support for Immediate SACK has been confirmed for the association, see Section 4, and the SCTP sender itself deploys the Immediate SACK feature.

For further details on the TLPP transmission see Section 3.3.3. Observe that in some special cases no TLPP is sent even if this state is entered and conceptually is handled as if a TLPP has been sent.

In the PROBE WAIT state the following may happen:

- o SACKs may arrive that makes the miss indication count on the lowest outstanding TSN/lowest TSN in flight reach Dupthresh in which case the PROBE WAIT state is terminated and Fast Recovery is started.
- o A SACK cumulatively acknowledging all holes including the lowest outstanding TSN may bring the SCTP TLR STM state back to SCTP TLR OPEN state. In this case a new PTO timer will be started on the new lowest outstanding TSN following the PTO timer setting in the SCTP TLR OPEN state. In this situation "PTO restart principles" (i.e., yielding $\text{PTO} - \text{T_latest}(\text{TSN})$) shall not be deployed. Spurious entering of PROBE WAIT state can happen if the PTO is too short, in such a situation it would not be prudent to deploy PTO restart principles when returning to OPEN state. OPEN ISSUE: Possibly PTO restart principles shall be refrained from until new RTT measurements are available.
- o A SACK may arrive for a higher outstanding TSN with lowest outstanding TSN on the path remaining unSACK'ed. This will result in declaration of the packet of the lowest outstanding TSN as lost and will make SCTP enter Fast Recovery.
- o A SACK may arrive that acknowledges the lowest outstanding TSN, but also data of higher TSN than the new lowest outstanding TSN are acknowledged in the SACK. In this case there is indication that either packet re-ordering has occurred or the new lowest outstanding TSN has been lost. The state will now transit to SCTP TLR DELAY WAIT state for potential entering of SCTP TLR driven Fast Recovery if the PTO timer kicks prior to the new lowest outstanding TSN has been acknowledged. The PTO timer set on the

new lowest outstanding TSN will be $PTO2 - T_latest(TSN)$. In case $PTO2 - T_latest(TSN) \leq 0$, the DELAY WAIT state is immediately terminated, the packet containing the lowest outstanding TSN is declared lost, and Fast Recovery is started.

- o The T3-timer may kick. In this case the PROBE WAIT state will be terminated and T3-recovery will start on non-SACK'ed outstanding data.

3.3.2.3. SCTP TLR DELAY WAIT STATE

In this state network responsiveness has been received (in form of a SACK of higher TSN than the lowest outstanding TSN) and the PTO timer relative to the lowest outstanding TSN is running for potential entering of SCTP TLR driven Fast Recovery.

The PTO set on a new lowest outstanding TSN in this state will be according to $PTO2$ in form of $PTO2 - T_latest(TSN)$.

In the DELAY WAIT state the following may happen:

- o SACKs may arrive that will make the miss indication count on the lowest TSN in flight reach Dupthresh, the DELAY WAIT state is terminated and SCTP enters Fast Recovery.
- o The PTO timer relative to the lowest outstanding TSN may kick. This will result in declaration of packet of the lowest outstanding TSN as lost and will make SCTP enter Fast Recovery.
- o A SACK cumulatively acknowledging all holes including the lowest outstanding TSN may arrive and bring the SCTP TLR STM state back to SCTP TLR OPEN state and the PTO timer will be restarted on the new lowest outstanding TSN. The PTO timer value set will be the value $PTO - T_latest(TSN)$ where the PTO value is calculated either from $PTO1$ or $PTO2$ according to the evaluation criteria given for the OPEN state.
- o A SACK may arrive that acknowledges the lowest outstanding TSN, but also data of higher TSN than the new lowest outstanding TSN are acknowledged in the SACK. In this case there is indication that either packet re-ordering has occurred or the new lowest outstanding TSN has been lost. The state will remain in SCTP TLR DELAY WAIT state for potential entering of SCTP TLR driven Fast Recovery if the PTO timer kicks prior to the new lowest outstanding TSN has been acknowledged. The PTO timer set on the new lowest outstanding TSN will be $PTO2 - T_latest(TSN)$. In case $PTO2 - T_latest(TSN) \leq 0$, the DELAY WAIT state is terminated, the

packet containing the lowest outstanding TSN is declared lost and Fast Recovery is started.

- o A SACK may arrive that does not acknowledge the lowest outstanding TSN and still do not make the miss indication count reach the Dupthresh value. In this situation no changes are done to the PTO timer running and the state will remain in SCTP TLR DELAY WAIT state for potential entering of SCTP TLR driven Fast Recovery if the PTO timer kicks prior to the lowest outstanding TSN has been acknowledged.

3.3.2.4. Exit of Loss Recovery

After exit of Fast Recovery or completion of T3-retransmission then if data is outstanding a PTO timer is started relative to the lowest outstanding TSN on the path and the state transits to either SCTP TLR OPEN state or to SCTP TLR DELAY Wait state depending on the status of the SACK scoreboard (i.e., do gaps exist or not). The PTO timer set will follow the rules described above. PTO-restart principles shall not be deployed in this situation as fresh RTT measurements might not be available. OPEN ISSUE: Possibly PTO restart principles shall be refrained from until new RTT measurements are available.

3.3.2.5. RTO-Restart Principles for the T3-timer

When the lowest TSN in flight on a path is undergoing Fast Recovery or T3-retransmission a T3-timer is running on the path (relative to this lowest TSN in flight). For SCTP TLR the RTO-restart principles as of [HURTIG] SHOULD unconditionally be applied to the T3-timer. Thus the T3-timer set on a path in this case SHOULD be the value RTO-T_latest(TSN) relative to the lowest TSN in flight on the path.

3.3.3. TLPP Transmission Rules

The transmission of a Tail Loss Probe Packet (TLPP), done just prior to entering the SCTP TLR PROBE WAIT state from SCTP OPEN, is governed by the following details:

- o TLPP of new data is always preferred if such is available for transmission. If such exists, the TLPP sent is chosen as the lowest unsent TSNs that fit into one packet
- o Alternatively if no new data is available for transmission, either due to application or receiver side limitations, the presently outstanding packet with highest TSN number is chosen as the TLPP.
- o TLPP of retransmission data counts twice in the in-flight until acknowledged or detected as lost.

- o The transmittal of a TLPP of sub-PMTU size is not blocked by Nagle-like bundling.

The highest (new) outstanding TSN is chosen for probing in order to best possibly interface with standard Fast Recovery, i.e., to create a loss pattern situation that corresponds best possibly with how Fast Recovery algorithm retransmits, and is invoked to retransmit, lost packets.

TLPP Transmission conditions:

A TLPP is not sent unconditionally when SCTP enters PROBE WAIT state on a path.

No explicit limit is applied to the number of TLPP probe packets (i.e., the number of unacknowledged packets sent as TLPP) that may be outstanding at any given time but the number of such will in most situations be effectively limited to a very few (very often only one) by the following rules based on latency and congestion control principles; Generally a TLPP will not be allowed to breach the CWND more than once per RTT and further a TLPP is omitted to be sent if an already outstanding packet is considered to serve "good enough" from a network probing perspective. In addition special considerations are given for the transmittal of a TLPP consisting of retransmission data to ease loss masking detection (see Section 3.3.4). It is further noted that the frequency of TLPP transmittal is limited by how often a transition can happen out of and back into the PROBE WAIT state.

The conditional transmission of a TLPP is specified as follows:

- o If the highest outstanding TSN has been sent only a little while ago, this TSN effectively serves as a probe and no TLPP need to be send. This condition aims to prevent unnecessary retransmission of just sent data and unnecessary transmittal of small sub-PMTU packets of new data. The exact condition to apply is:

* If $T_Latest(highTSN) < \gamma * SRTT$

then no TLPP is sent. $\gamma = 1/2$ is recommended. A special condition arise when little data is outstanding and the SACK of the outstanding data may be lost by a single loss of SACK. In this case the transmittal of a TLPP packet will make the SACK return be robust toward single loss of SACK. For added robustness to SACK return an SCTP TLR implementation MAY disregard the above condition if only 2 packets are outstanding.

- o If no TLPP is outstanding, a probe is sent unconditionally of CWND.
- o If a TLPP is outstanding, a probe is sent conditionally to that there is room in CWND. Otherwise no TLPP is sent. I.e., the CWND is not breached when a TLPP is outstanding.
- o If no new data exists, a probe of retransmission data is sent conditional to whether a TLPP of retransmission data is already outstanding. I.e.,:
 - * If no TLPP of retransmission data is outstanding, send TLPP consisting of highest outstanding TSN.
 - * If a TLPP of retransmission data is outstanding, no TLPP is sent.

The above rules on probes of retransmission data are defined to ease the detection of TLPP recovered losses by the algorithm described in Section 3.3.4.

3.3.3.1. Multi-Path Considerations for TLPP Transmission

In multi-homed [RFC4960] SCTP, multiple paths may have a PTO timer running on data in flight. E.g., two paths may be in SCTP OPEN state and SCTP will have two PTO timers running, each relative to the lowest outstanding TSN on the respective path. This (exception) situation in particular can occur as a result of a change of the data transfer path as a result of a switchback operation to a primary path. The handling of TLPP transmission for SCTP MH is described in the following. The underlying philosophy of the solution is, as far as possible, to have the SCTP TLR probing mechanism be undertaken on, and by, the data transfer path. Thus best possibly avoiding conflicts that may arise due to concurrent data transfers on multiple paths. As follows:

- o When the PTO timer kicks on a path in SCTP OPEN state and the TLPP selected by the rules above consists of new data, then if the path is the present data transfer path of the association the TLPP will be sent and in this case the TLPP is sent on the data transfer path of the association. When in this situation the path is not the present data transfer path of the association, then
 - * if there is no outstanding data on the present data transfer path, the TLPP of new data is sent there.
 - * if there is outstanding data on the data transfer path, the TLPP is not sent. Instead the potential transmittal of a TLPP

is deferred to be driven by a later kick of the PTO timer on the data transfer path.

The first situation that data is available for transmittal on the data transfer path but has not been sent, is an unlikely situation, but it might possibly occur in some implementations.

- o When the PTO timer kicks on a path in SCTP OPEN state and the TLPP selected by the rules above consist of retransmission of the presently highest outstanding TSNs on the association, then if and only if these TSNs are outstanding on the path in question is the TLPP allowed to be sent. The following guidelines are given for the path selection for the TLPP:
 - * An SCTP implementation which does not implement the Unambiguous SACK feature of Appendix A should send the TLPP on the path on which the TSNs are presently outstanding (i.e., on the path on which the PTO kicked).
 - * An SCTP implementation which implements the Unambiguous SACK feature of Appendix A may send the TLPP on the data transfer path of the association.

The reason a TLPP of retransmitted data in the first case above is sent on the path on which the data was first sent, even if this path is not the present data transfer path (special corner case with change of data transfer path or destination address directed data transfer), is that the TLPP Loss Mask Detection mechanism, see Section 3.3.4 could not infer on which path to perform a congestion window reduction if the TLPP and original data is sent on different paths. An SCTP implementation which implements the Unambiguous SACK feature of Appendix A can better distinguish the SACK of the original TSN and the retransmitted TSN and can therefore operate differently. The choice of sending the TLPP on the data transfer path may be motivated by that the Fast Recovery procedure, which the SACK of the TLPP may result in, would use the data transfer path. On the other hand then differences in the RTT on the different paths may make it suboptimal to send the TLPP on the data transfer path as well as it can give rise to potential uncertainty in the TLPP Loss Recovery Mask detection and reaction process (see Section 3.3.4).

It is emphasized that the deferral of the transmission of a TLPP does not prevent entering of the PROBE WAIT state on the path where the PTO kicked.

3.3.4. Masking of TLPP Recovered Losses

If a single SCTP packet is lost, there is a risk that the TLPP packet itself might repair the loss if that particular lost packet is used as probe. The masking problem is only present if the TLPP is based on retransmission data. The TLPP might mask the loss and thus interfere with the congestion control principle that requires for CWND halving when a loss is detected.

At present the solution in this document operates with the algorithm defined for this purpose in [DUKKIPATI01] with adjustment to SCTP to rely on the D-SACK (duplicate TSN received) information available from SCTP SACK or alternatively to the information available from the Unambiguous SACK information of Appendix A. The solution operates with a conceptual TLPP Retransmission Episode. As follows:

- o Once a TLPP packet consisting of retransmission data is sent a TLPP Retransmission Episode is started.
- o A TLPP Retransmission Episode is abruptly terminated if Fast Recovery or T3-Recovery is entered.
- o For an SCTP implementation which does not implement the Unambiguous SACK feature of Appendix A, as well as for an SCTP association where the Unambiguous SACK feature of Appendix A is not in use, the TLPP Retransmission Episode terminates when an incoming SACK cumulatively acknowledges a sequence number higher than the sequence number of the TLPP probe with retransmission data. If at this time in stage the number of times the TLPP TSN has been received, according to the D-SACK information received, is lower than the number of times the TLPP TSN has been sent, CWND halving is done on the unique path on which the retransmission TLPP TSN has been sent. Further at this stage in time the contribution from the TSN is subtracted from the flight size in accordance to the number of times the TSN has been sent.
- o For an SCTP implementation which implements the Unambiguous SACK feature of Appendix A the following actions are taken at the time of acknowledgement of the TSN used as TLPP:
 - * If the TLPP TSN is first cumulatively acknowledged in a SACK with CUMACK TSN = TLPP TSN and with no SACK (or CUMACK) of higher TSNs, then from the Unambiguous SACK information SCTP sender can classify to be in the following cases:
 - + The original TSN has not (yet) been received, the retransmission TSN (the TLPP) has been received.

- In this case the original TSN is judged as lost, CWND halving is performed on the path on which the original TSN was sent and the sent TSNs are subtracted from the flight size(s). This concludes the TLPP Retransmission Episode.
- + Both the original transmission as well as the retransmission (the TLPP) have been received.
 - In this case the sent TSNs are subtracted from the flight size(s). This concludes the TLPP Retransmission Episode.
- + The original TSN has been received, the retransmission TSN (the TLPP) has not yet been received:
 - In this case a special timer is started with value $PTO - T_latest(TSN)$ and the bytes of the retransmitted TSN (the TLPP) remains in the flightsize of the path on which it was sent until either of the following happens - whichever happens first:
 - o Unambiguous SACK of the TSN is received in which case the TSN is subtracted from the flightsize and the timer is stopped. This concludes the TLPP Retransmission Episode.
 - o A SACK of a higher TSN than the TLPP arrives with unambiguous SACK information indicating that the TLPP has not been received. Now marking is made on the path so that, if when the timer kicks, the TSN has still not been acknowledged, the TSN is judged as lost, CWND halving is done and the TSN is subtracted from the flightsize. This then concludes the TLPP Retransmission Episode.
 - o The timer kicks, the TSN is subtracted from the flightsize (but no CWND halving is done). This concludes the TLPP Retransmission Episode.
- * If the TLPP TSN is first cumulatively acknowledged in a SACK with highest SACK'ed (or CUMACK'ed) $TSN > TLPP\ TSN$, then from the Unambiguous SACK information SCTP sender can classify the same cases as above and take corresponding actions. One additional situation can arise in this situation:
 - + Only one of the transmissions of the TSN has been received, but no clear Unambiguous SACK indication of which that was received is available from the SACK. This uncertainty can

only result from situations where SACKs are lost, potentially in combination with that more data chunks than the TSN it self were outstanding at the time when the TLPP was sent and some of this data arrived later at the receiver than the original TSN or the TLPP.

- In this case the original TSN is judged as having been received and it is subtracted on the flightsize of the path on which it was sent. The timer PTO-T_latest(TSN) is set and handling of potential CWND reduction caused by loss of the TLPP is handled following the principles described above.

DISCUSSION of Unambiguous SACK Case Handling: CWND halving is not prescribed to be done for a potential lost retransmitted TSN used as TLPP in all cases above as there is no guarantee that a SACK confirming a potential arrival of the retransmitted TSN will arrive in time (i.e., this SACK may be lost). CWND halving is done if SACK of a higher TSN number than the TLPP number has arrived, PTO time has elapsed since the transmittal of the TLPP and the TLPP in it self cannot be determined to be received from the Unambiguous SACK information.

3.3.5. Elimination of unnecessary DELAY-ACK delays

The negative impact of DELAY_ACK on the loss recovery delay is partially mitigated by setting of the I-bit on TLPP.

OPEN ISSUES:

- o It is to be determined if the Immediate SACK feature shall be relied on more aggressively. Possible options are:
 - * Immediate SACK flag to be set on all retransmitted TSNs.
 - * Immediate SACK flag to be set on all TSNs that are sent where the transmittal of an immediate following subsequent packet cannot be foreseen. This effectively would result in that the I-bit is set on a sent TSN whenever either of the following is true:
 - + no more chunks can be sent right after this chunk due to CWND limitations.
 - + no more chunks can be sent right after this due to RCV window limitations

- + no more chunks can be sent right after this as no more chunks are available in the SND buffer.
- + no more chunks can be sent right after this due to Nagle.
(May depend on the exact Nagle-like implementation).

For the second choice it would be relevant to use PTO1 setting for the PTO timer on all TSNs sent with the I-bit set, when the receiver is known to support the Immediate SACK feature. The downside of this choice is that it very severely limits the effectiveness of the DELAY_ACK feature.

- o Ideally the PTO timer relative to the lowest outstanding TSN should be adjusted to follow PTO2 when a subsequent packet is transmitted. The downside of this choice is the implementation impacts of such detailed - potentially per packet transmission - logic. To be elaborated further.

4. Confirmation of support for Immediate SACK

Confirmation of receiver support of the Immediate SACK function, [RFC7053] is established by an SCTP TLR sender by the following means:

- o In case the data chunk of [RFC4960] is in use on the association, confirmation of [RFC7053] support by the SCTP receiver is assumed if SCTP TLR sender receives a data chunk with the I-bit flag set.
- o [TO DE CONFIRMED:] In case the I-data chunk of [SCTP-IDATA] is in use on the association, SCTP sender can by [SCTP-IDATA] assume that SCTP receiver supports [RFC7053].

5. Socket API Considerations

This section will describe how the socket API defined in [RFC6458] is extended to provide a way for the application to control the retransmission algorithms in operation in the SCTP layer.

Socket option for control of the features is yet to be defined.

Please note that this section is informational only.

6. Security Considerations

There are no new security considerations introduced by the functions defined in this document.

7. Acknowledgements

The author acknowledges Henrik Jensen for his very significant contribution for the definition of, the implementation of and the experiments with function.

The work heavily draws on prior art work done for TCP, [DUKKIPATI01] in particular. The contributors of that work should be credited for many of the ideas put forward here for SCTP.

8. IANA Considerations

This document does not create any new registries or modify the rules for any existing registries managed by IANA.

9. Discussion and Evaluation of function

Experiments in progress. Details to be filled in.

Right now we use this section to retain a number of issues that are to further elaborated on:

- o A significant number of spurious TLR probes have been observed in tests. It is to be determined if this is a fact of the function or whether it may be improved with adjustment of the PTO timer calculations.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<http://www.rfc-editor.org/info/rfc4960>>.
- [RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <<http://www.rfc-editor.org/info/rfc5061>>.

- [RFC5062] Stewart, R., Tuexen, M., and G. Camarillo, "Security Attacks Found Against the Stream Control Transmission Protocol (SCTP) and Current Countermeasures", RFC 5062, DOI 10.17487/RFC5062, September 2007, <<http://www.rfc-editor.org/info/rfc5062>>.
- [RFC7053] Tuexen, M., Ruengeler, I., and R. Stewart, "SACK-IMMEDIATELY Extension for the Stream Control Transmission Protocol", RFC 7053, DOI 10.17487/RFC7053, November 2013, <<http://www.rfc-editor.org/info/rfc7053>>.
- [SCTP-IDATA] R. Stewart et al, , "Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol draft-ietf-tsvwg-sctp-ndata-04.txt", IETF Work In Progress , 07 2015.

10.2. Informative References

- [CARO01] A. Caro et al, , "Retransmission Policies with Transport Layer Multihoming", ICON , 2003.
- [CARO02] A. Caro et al, , "Retransmission Schemes for End-to-end Failover with Transport Layer Multihoming", GLOBECOM , 11 2004.
- [CMT-SCTP] Amer et al., P., "Load Sharing for the Stream Control Transmission Protocol (SCTP) draft-tuexen-tsvwg-sctp-multipath-10.txt", IETF Work In Progress , 5 2015.
- [DUKKIPATI01] Dukkupati, N., Cardwell, N., Cheng, Y., and M. Mathis, "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail", Work Expired , 2 2013.
- [DUKKIPATI02] Dukkupati, N., Mathis, M., Cheng, Y., and M. Ghobadi, "Proportional Rate Reduction for TCP", Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement , 11 2011.
- [HURTIG] P. Hurtig et al., , "TCP and SCTP RTO Restart, draft-ietf-tcpm-rtorestart-08", IETF Work In Progress , 3 2015.
- [MATHIS] Mathis, M., "FACK", ACM SIGCOMM Computer Communication Review 26,4, 10 1996.

- [Rajiullah] M. Rajiullah et al., , "An Evaluation of Tail Loss Recovery Mechanisms for TCP", ACM SIGCOMM Computer Communication Review 45,1, 1 2015.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <<http://www.rfc-editor.org/info/rfc3758>>.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<http://www.rfc-editor.org/info/rfc5681>>.
- [RFC5827] Allman, M., Avrachenkov, K., Ayesta, U., Blanton, J., and P. Hurtig, "Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)", RFC 5827, DOI 10.17487/RFC5827, May 2010, <<http://www.rfc-editor.org/info/rfc5827>>.
- [RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <<http://www.rfc-editor.org/info/rfc6458>>.
- [RFC6675] Blanton, E., Allman, M., Wang, L., Jarvinen, I., Kojo, M., and Y. Nishida, "A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP", RFC 6675, DOI 10.17487/RFC6675, August 2012, <<http://www.rfc-editor.org/info/rfc6675>>.
- [SCTP-PF] Y. Nishida et al., , "SCTP-PF: Quick Failover Algorithm in SCTP, draft-ietf-tsvwg-sctp-failover-13.txt", IETF Work In Progress , 09 2015.
- [zimmermann01] Zimmermann, A., "CUBIC for Fast Long-Distance Networks, draft-ietf-tcpm-cubic-00", IETF Work In Progress , 6 2015.
- [zimmermann02] Zimmermann, A., "The TCP Echo and TCP Echo Reply Option, draft-zimmermann-tcpm-echo-option-00", IETF Work In Progress , 6 2015.

[zimmermann03]

Zimmermann, A., "Using the TCP Echo Option for Spurious Retransmission Detection, draft-zimmermann-tcpm-spurious-rxmit-00", IETF Work In Progress , 7 2015.

Appendix A. Unambiguous SACK

When receiving a SACK of a TSN it is not possible to unambiguously determine if the receiver hereby acknowledges the first transmission of the TSN or possible subsequent retransmissions of the TSN, when such multiple transmissions of the same TSN have been made. The duplicate TSN information in the SCTP SACK chunk does help to provide information about how many times the same TSN has been received at the received side, but still it is not possible to unequivocally link the SACK information to the different transmissions of the same TSN. An additional source of ambiguity comes from the fact that packets may be duplicated in the network.

Unambiguous SACK information is generally beneficial for many SCTP protocol aspects, e.g., for improved RTT measurements, for more accurate loss detection, maintain of flightsize and congestion control operation.

Providing full accurate SACK information from receiver to sender side requires a reliable (and ordered) SACK feedback channel thus overcoming the information gap that may arise from loss (or from re-ordering) of SACKs. The establishment of such a reliable feedback Channel is not proposed but the proposal implements measures that allow for some robustness towards information loss due to SACK loss.

NOTE for AUTHORS: The solution is independent from a potential split of the SACK TSN Gap information in SACK and NR-SACK gaps respectively following [CMT-SCTP].

A.1. TSN Retransmission ID in Data Chunk Header

It is a prerequisite that the SCTP association deploy, and has negotiated usage of, the new I-data chunk of [SCTP-IDATA].

We define a new 4-bit Retransmission ID (RTX ID) in the I-data Chunk header. The 4 bits consume 4 bits of the new reserved 16-bit field of the I-data chunk header. See Figure 1.

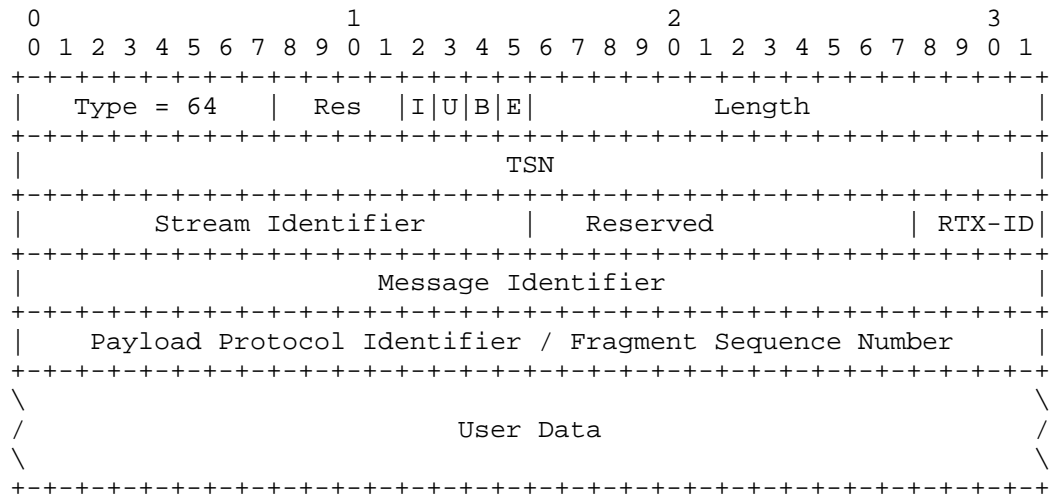


Figure 1: RTX-ID in I-DATA chunk format

A.1.1. Sender side behaviour

New data MUST be sent with RTX-ID =0. Whenever SCTP retransmits a data chunk it SHOULD step up the RTX ID. The highest RTX ID = 15 is used for all retransmissions of the same TSN beyond the 15-th retransmission or when the RTX ID last used for this TSN is 15. An SCTP sender MAY step the RTX ID up with more than one count when retransmitting a TSNs in order to have all TSNs within the SCTP packet use the one and the same RTX ID.

A.1.2. Receiver side behaviour

An SCTP receiver supporting this feature MUST process the RTX ID for all received TSNs in accordance with the prescriptions for Unambiguous SACK return below.

A.2. Unambiguous SACK Chunk

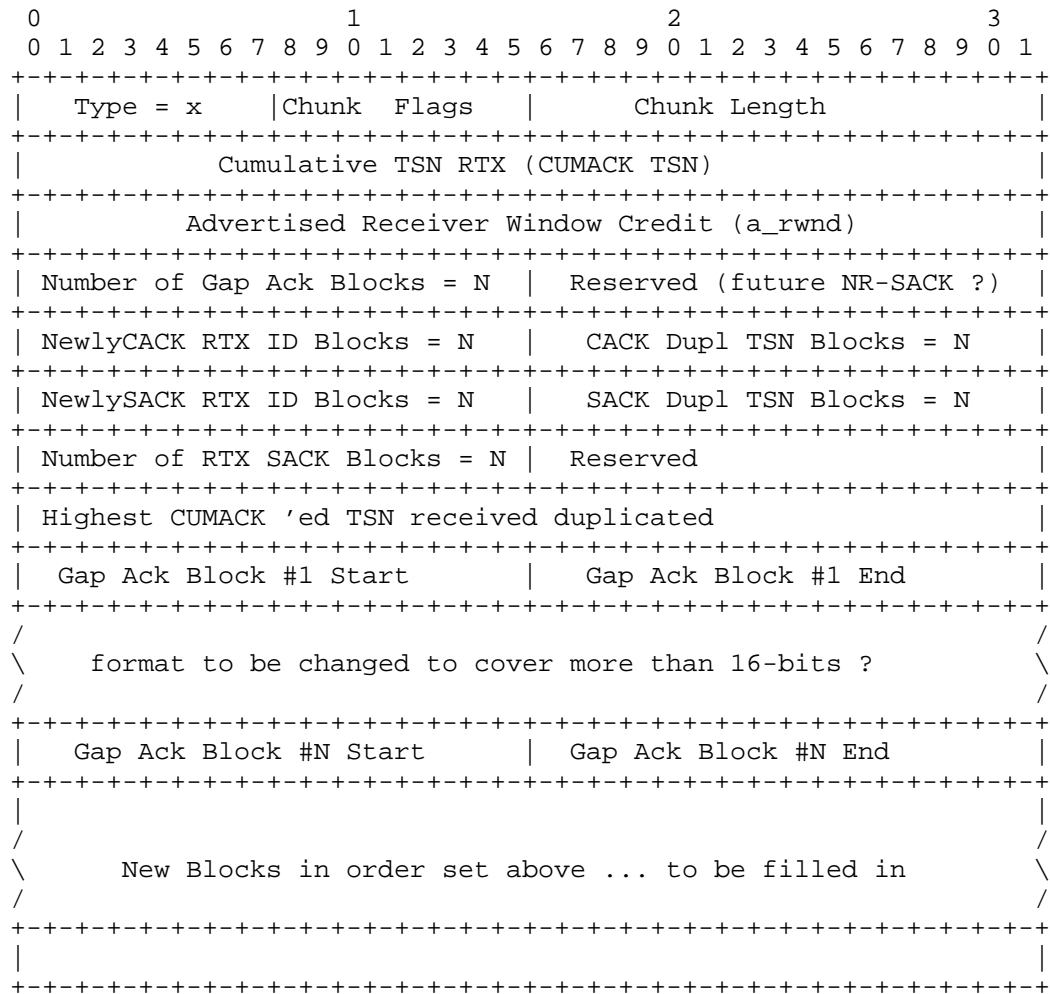


Figure 2: Unambiguous SACK chunk format

Newly CACK RTX ID block:

This block provides information on the newly acknowledged TSNs that were cumulatively acked in this SACK and for which the following hold:

- * The TSN is newly acked in this SACK. I.e., the TSN has not been received before (or if it has been received before it was since reneged).

- * The newly acknowledged TSN was received with RTX ID different from zero.

The RTX ID received with the TSN is returned in this block. The information returned in a CACK RTX ID block is a consecutive range of TSN fulfilling the above for which identical RTX ID has been received. Proposed format is off-set from CUMACK TSN (lower than CUMACK TSN), length of range and RTX ID.

Newly SACK RTX ID block:

This block provides information on the newly acknowledged TSNs that were selectively acknowledged in this SACK and for which the following hold:

- * The TSN is newly acked in this SACK. I.e., the TSN has not been received before (or if it has been received before, it was since reneged).
- * The newly acknowledged TSN was received with RTX ID different from zero.

The RTX ID received with the TSN is returned in this block. The information returned in a SACK RTX ID block is a consecutive range of TSN fulfilling the above for which identical RTX ID has been received. Proposed format is off-set from CUMACK TSN (higher than CUMACK TSN), length of range and RTX ID - OR alternatively format of present SACK blocks with off set bounded by 16-bit to CUMACK TSN.

Newly CACK Dupl TSN block:

This block provides information on the TSNs received since last returned SACK for which following hold:

- * The TSN is lower than or equal to the CUMACK TSN.
- * The TSN is a duplicate. Meaning that a data chunk with same TSN, but possibly different RTX ID, has been received.

The RTX ID received with the TSN is returned in this block. The information returned in a CACK Dupl TSN block is a consecutive range of TSN fulfilling the above for which identical RTX ID has been received. Proposed format is off-set from CUMACK TSN (lower than CUMACK TSN), length of range and RTX ID. The RTX ID may be zero.

Newly SACK Dupl TSN block:

This block provide information on the TSNs received since last returned SACK for which the following hold:

- * The TSN is higher than the CUMACK TSN.
- * The TSN is a duplicate. Meaning that a data chunk with same TSN, but possibly different RTX ID, has been received.

The RTX ID received with the TSN is returned in this block. The information returned in a SACK Dupl TSN block is a consecutive range of TSN fulfilling the above for which identical RTX ID has been received. Proposed format is off-set from CUMACK TSN (higher than CUMACK TSN), length of range and RTX ID - OR - format of present SAC blocks with off set bounded by 16-bit to CUMACK TSN. The RTX ID may be zero.

Together with the existing SACK information, the Newly CACK/SACK RTX ID and the CACK/SACK Dupl TSN blocks provide unambiguous SACK information for all received TSNs differentiating on the RTX ID received with the TSN. The information may be partially lost from the receiver to the sender if a SACK is lost. The RTX SACK Block and the Highest CUMACK Received Duplicated information is returned in order to provide means to recover part of the information that can be lost when a SACK is lost.

RTX SACK block:

This block provides information on the TSNs for which the following hold:

- * The TSN has been received and has been selectively acked in prior SACKs (OPEN: alternatively in SACKs including this one).
- * The TSN is higher than the CUMACK TSN.
- * The TSN has been received only with RTX IDs different from zero.

The information returned in an RTX block is a consecutive range of TSN fulfilling the above. Proposed format is off-set from CUMACK TSN (higher than CUMACK TSN) and length of range - OR - format of present SACK blocks with off set - bounded by 16-bit to CUMACK TSN.

Highest CUMACK'ed TSN received Duplicated:

Here the highest TSNs that fulfill the following condition is inserted:

- * The TSN has been received duplicated
- * The TSN is lower than or equal to the CUMACK TSN.

When no duplicates have been seen or when no duplicates have been seen in last 2^{31} window of TSNs that have been cumulatively acknowledged, CUMACK TSN +1 is returned.

By means of the RTX SACK block an SCTP sender may recover the information that a SACK'ed TSN does not represent the original TSN first sent. I.e., the TSN sent with RTX ID = 0.

By means of the "Highest CUMACK'ed TSN received Duplicated" an SCTP receiver may recover the information that more than one incarnation of a TSN has been received when the SACK, which cumulatively acknowledged the arrival of the different incarnations of the TSN, in it self was lost. The particular example of special interest is the case where the one and the same SACK would contain information on receipt of both the original TSN and a spurious retransmission of the TSN. Such can happen in scenarios where DELAY_ACK handling at the receiver side delays the return of SACK information and a SACK is lost, even if the original data and the spurious retransmission data was sent with reasonable spacing in time.

A.2.1. Receiver side behaviour

The RTX SACK Block and the Highest CUMACK information to be returned in SACKs demand for an SCTP receiver to keep track (state) of the following information on a per association basis:

- o A list (or ranges) of TSNs that have been SACK'ed, but not yet cumulatively acknowledged and for which RTX ID = 0 has not been seen. It is noted that the TSN data chunk itself may have been delivered to the application.
- o The highest TSN lower than CUMACK TSN for which a duplicate has been received.

A.3. Unambiguous SACK return

Whenever Unambiguous SACKs are in use on an association and SCTP receives a valid data chunk with RTX-ID different from zero it shall not delay the return of the Unambiguous SACK. Otherwise Unambiguous SACKs are returned at any time when an [RFC4960] implementation would return a SACK.

A window opener MUST include Unambiguous SACK information.

A.4. Negotiation

An SCTP receiver MUST NOT send an Unambiguous SACK chunk unless both peers have indicated its support of the Unambiguous SACK feature within the Supported Extensions Parameter as defined in [RFC5061]. If Unambiguous SACK has been negotiated on an association, Unambiguous SACKs MUST be returned whenever a SCTP receiver would return SACK information. If Unambiguous SACK has not been negotiated on an association, the RTX-ID field in the chunk header of incoming data chunks MUST be ignored and [RFC4960] SACK format and return policies MUST be adhered to.

Authors' Addresses

Karen E. E. Nielsen
Ericsson
Kistavaegen 25
Stockholm 164 80
Sweden

Email: karen.nielsen@tieto.com

Rafaelle De Santis
Ericsson
xx
xx xx
Italy

Email: rafaele.de.santis@ericsson.com

Anna Brunstrom
Karlstad University
Universitetsgatan 2
Karlstad 651 88
Sweden

Email: anna.brunstrom@kau.se

Michael Tuexen
Muenster Univ. of Appl. Science
Stegerwaldstrasse 39
Steinfurt 48565
Germany

Email: tuexen@fh-muenster.de

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

Email: randall@lakerest.net

Internet Engineering Task Force
INTERNET-DRAFT
Intended Status: Standards Track
Expires: January 2, 2016

X. Wei
L.Zhu
Huawei Technologies
L.Deng
China Mobile
B.Briscoe
July 1, 2015

Tunnel Congestion Feedback
draft-wei-tsvwg-tunnel-congestion-feedback-04

Abstract

This document describes a mechanism to calculate congestion of a tunnel segment based on RFC 6040 recommendations, and a feedback protocol by which to send the measured congestion of the tunnel from egress to ingress. A basic model for measuring tunnel congestion and feedback is described, and a protocol for carrying the feedback data is outlined.

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) 2015 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	3
3. Congestion Information Feedback Models	4
3.1 Direct Model	4
3.2 Centralized Model	4
4. Congestion Level Measurement	5
5. Congestion Information Delivery	7
5.1 IPFIX Extentions	7
5.1.1 ce-cePacketTotalCount	7
5.1.2 ect-nectPacketTotalCount	8
5.1.3 ce-nectPacketTotalCount	8
5.1.4 ce-ectPacketTotalCount	8
5.1.5 ect-ectPacketTotalCount	9
6. Congestion Management	9
7. Security	9
8. IANA Considerations	10
9. References	10
9.1 Normative References	10
9.2 Informative References	10
Authors' Addresses	11

1. Introduction

In IP network, persistent congestion (or named congestion collapse) would cause transport throughput to drop down, lead to waste of network resource, so appropriate congestion control mechanisms are critical to make sure the network not fall into persistent congestion state. Currently, transport protocols such as TCP, SCTP, DCCP, has their built-in congestion control mechanism, and even for certain single transport protocol like TCP there could be a couple of different congestion control mechanism to choose. 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, e.g., (1) some protocol's congestion control scheme might has internal design flaws; (2) improper software implementation of protocol; (3) some transport protocols even don't provide congestion control at all.

In order to have a better control on network congestion status, it's necessary for the network side to do certain kind of traffic control. For example, ConEx [ConEx] provides a method for network operator to learn about traffic's congestion contribution information, and then congestion management action could be taken based on this information.

Tunnels are widely deployed in various networks including public Internet, datacenter network, and enterprise network etc, a tunnel consists of an ingress, an egress and a set of interior routers. For the tunnel scenario, a tunnel-based mechanism which is different from ConEx is introduced for network traffic control to keep network away from persistent congestion; in this case, tunnel ingress will implement congestion management function to control the traffic entering the tunnel.

In order to do congestion management at ingress, the ingress must first get the inner tunnel congestion level information. But the ingress cannot use the locally visible traffic rates, because it would require additional knowledge of downstream capacity and topology, as well as cross traffic that does not pass through this ingress.

This document provide a mechanism of feeding back inner tunnel congestion level to ingress, using this mechanism the egress could feed the tunnel congestion level information it collects back to ingress, after receiving the information ingress could do congestion management according to network management policy.

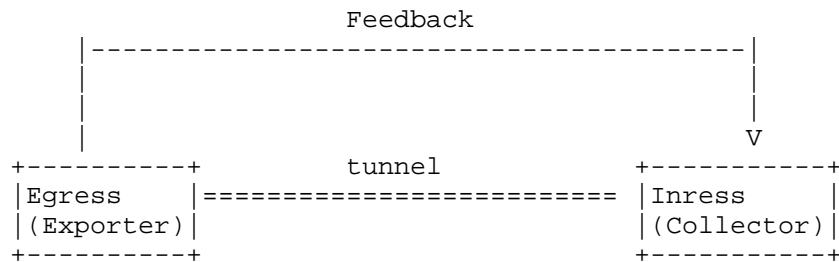
2. Conventions

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 RFC 2119 [RFC2119]

3. Congestion Information Feedback Models

According to specific network deployment, there are two kinds of feedback model: direct model and centralized model.

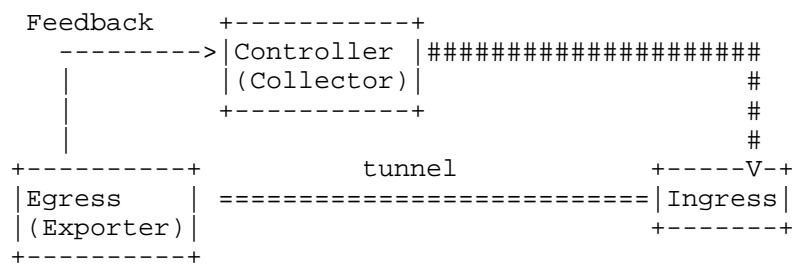
3.1 Direct Model



(a) Direct Feedback Model.

Direct model means egress feeds information directly to ingress. In this model, egress collects network congestion level information and feedback the information to ingress for congestion management. The ingress here will act as both decision point that decides how to do congestion management and action point that implements congestion management decision.

3.2 Centralized Model



(b) Centralized Feedback Model

There are scenarios that ingress only takes the role of action point, and it implements traffic control decision from another entity, named "controller" here.

In this model, after egress collects network congestion level information, it feeds back the information to controller instead of ingress, and then the controller makes congestion management decision and sends the decision to ingress.

4. Congestion Level Measurement

This section describes how to measure congestion level in tunnel.

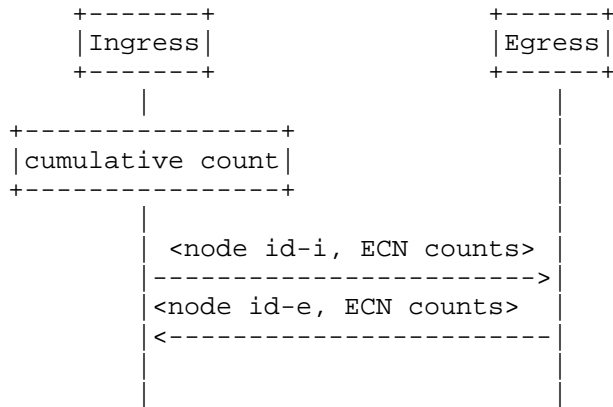
There may be different approaches of packet loss detection for different tunneling protocol scenarios, for instance, if there is a sequence field in tunneling protocol header, it will be easy for egress to detect packet loss through the gaps in sequence number space; another approach is to compare the number of packets entering ingress and the number of packets arriving at egress over the same span of packets. This document will focus on the latter one which is a more general approach.

If the routers support ECN, after router's queue length is over a predefined threshold, the routers will mark ECN packets as CE packets or drop not-ECN packets with the probability proportional to queue length, if the queue overflows all packets will be dropped; if the routers don't support ECN, after router's queue length is over a predefined threshold, the routers will drop both ECN packets and not-ECN packets with the probability proportional to queue length. It's assumed all routers in the tunnel support ECN.

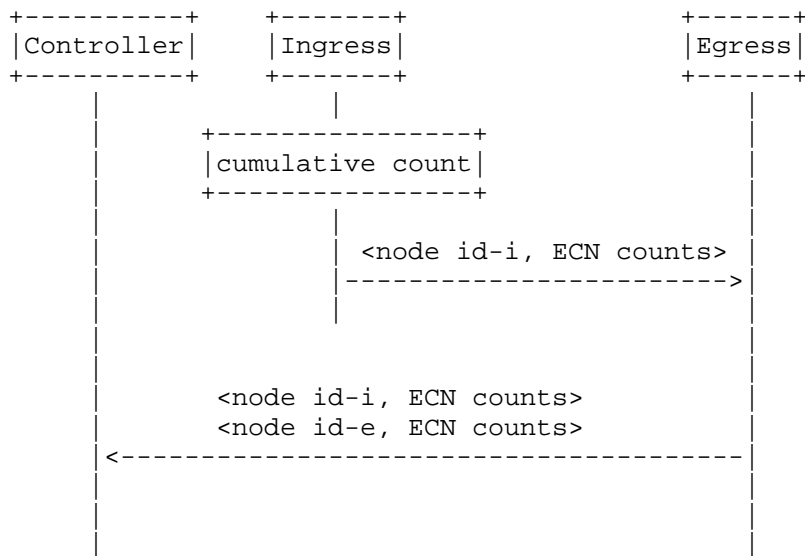
Faked ECT is used at ingress to defer packet loss to egress. The basic idea of faked ECT is that, when encapsulating packets, ingress first marks tunnel outer header according to RFC6040, and then remarks outer header of Not-ECT packet as ECT, there will be three kinds of combination of outer header ECN field and inner header ECN field: CE|CE, ECT|N-ECT, ECT|ECT (in the form of outer ECN| inner ECN).

In case all interior routers support ECN, the network congestion level could be indicated through the ratio of CE-marked packet and the ratio 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, because loss is only ever a sign of serious congestion, so it doesn't need to measure loss ratio accurately.

The basic procedure of congestion level measurement is as follows:



(a) Direct model feedback procedure



(b) Centralized model feedback procedure

Ingress encapsulates packets and marks outer header according to faked ECT as described above. Ingress cumulatively counts packets for three types of ECN combination (CE|CE, ECT|N-ECT, ECT|ECT) and then the ingress regularly sends cumulative packet counts message of each type of ECN combination to the egress. When each message arrives, the

egress cumulatively counts packets coming from the ingress and adds its own packet counts of each type of ECN combination (CE|CE, ECT|N-ECT, CE|N-ECT, CE|ECT, ECT|ECT) to the message and either returns the whole message to the ingress, or to a central controller.

The counting of packets could 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; or 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 message of cumulative packet counts of each type of ECN combination to tunnel egress, and the tunnel egress also needs to feed the message of cumulative packet counts of each type of ECN combination to the ingress or central collector. This section describes how the messages could be conveyed.

The message could be along the same path with network data traffic, referred as in band signal; or go through a different path with network data traffic, referred as out of band signal. Because out of band scheme needs additional separate path which might limit its actual deployment, so the in band scheme will be discussed here.

Because the message is transmitted in band, so the message packet might get lost in case of network congestion. To cope with the situation that message packet gets lost, the packet counts values are sent as cumulative counters, so if a message is lost the next message will recover the missing information.

IPFIX [RFC7011] is selected as a choice of candidate protocol. IPFIX is preferred to use SCTP as transport, and because SCTP allows partially reliable delivery [RFC3758], which makes sure the feedback message will not be blocked to be sent in case of SCTP packets lost due to network congestion.

When sending message from ingress to egress, the ingress acts as IPFIX exporter and egress acts as IPFIX collector; when sending message from egress to ingress or controller, the egress acts as IPFIX exporter and ingress or controller acts as IPFIX collector.

5.1 IPFIX Extensions

5.1.1 ce-cePacketTotalCount

Description: The total number of incoming packets with CE|CE ECN

marking combination for this Flow 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: packets

5.1.2 ect-nectPacketTotalCount

Description: The total number of incoming packets with ECT|N-ECT ECN marking combination for this Flow 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: packets

5.1.3 ce-nectPacketTotalCount

Description: The total number of incoming packets with CE|N-ECT ECN marking combination for this Flow 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: packets

5.1.4 ce-ectPacketTotalCount

Description: The total number of incoming packets with CE|ECT ECN

marking combination for this Flow 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: packets

5.1.5 ect-ectPacketTotalCount

Description: The total number of incoming packets with ECT|ECT ECN marking combination for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD5

Statuses: current

Units: packets

6. Congestion Management

After tunnel ingress (or controller) 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.

Congestion management action must be delayed by more than a worst-case global RTT, 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 [CB] and congestion policing [CP] could be applied.

7. Security

This document describes the tunnel congestion calculation and

feedback. For feeding back congestion, security mechanisms of IPFIX are expected to be sufficient. No additional security concerns are expected.

8. IANA Considerations

This document defines a set of new IPFIX Information Elements (IE). New registry for these IE identifiers is needed.

TBD1~TBD5.

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, <<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, September 2001, <<http://www.rfc-editor.org/info/rfc3168>>.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004, <<http://www.rfc-editor.org/info/rfc3758>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, September 2013, <<http://www.rfc-editor.org/info/rfc7011>>.

9.2 Informative References

- [CONEX] Matt Mathis, Bob Briscoe. "Congestion Exposure (ConEx) Concepts, Abstract Mechanism and Requirements", draft-ietf-conex-abstract-mech-13, October 24, 2014
- [CB] G. Fairhurst. "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-01, April 02, 2015
- [CP] Bob Briscoe, Murari Sridharan. "Network Performance Isolation in Data Centres using Congestion Policing", draft-briscoe-

conex-data-centre-02, February 14, 2014

Authors' Addresses

Xinpeng Wei
Beiqing Rd. Z-park No.156, Haidian District,
Beijing, 100095, P. R. China
E-mail: weixinpeng@huawei.com

Zhu Lei
Beiqing Rd. Z-park No.156, Haidian District,
Beijing, 100095, P. R. China
E-mail: lei.zhu@huawei.com

Lingli Deng
Beijing, 100095, P. R. China
E-mail: denglingli@gmail.com

Bob Briscoe
B54/77, Adastral Park
Martlesham Heath
Ipswich IP5 3RE
UK