Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

A. Jain
A. Terzis
Google
H. Flinck
N. Sprecher
S. Arunachalam
Nokia Networks
K. Smith
Vodafone
V. Devarapalli
R. Bar Yanai
Vasona Networks
March 13, 2017

Mobile Throughput Guidance Inband Signaling Protocol
draft-flinck-mobile-throughput-guidance-04.txt

Abstract

The bandwidth available for end user devices in cellular networks can vary by an order of magnitude over a few seconds due to changes in the underlying radio channel conditions, as device mobility and changes in system load as other devices enter and leave the network. Furthermore, packets losses are not always signs of congestion. The Transmission Control Protocol (TCP) can have difficulties adapting to these rapidly varying conditions leading to inefficient use of a cellular network’s resources and degraded application performance. Problem statement, requirements and the architecture for a solution is documented in [Req_Arch_MTG_Exposure].

This document proposes a mechanism and protocol elements that allow the cellular network to provide near real-time information on capacity available to the TCP server. This "Throughput Guidance" (TG) information would indicate the throughput estimated to be available at the radio downlink interface (between the Radio Access Network (RAN) and the mobile device (UE)). TCP server can use this TG information to ensure high network utilization and high service delivery performance. The document describes the applicability of the proposed mechanism for video delivery over cellular networks; it also presents test results from live operator’s environment.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The problem statement related to the behavior of the TCP in cellular networks is provided in [Req_Arch_MTG_Exposure]. That same document specifies the requirements, reference architecture and proposed solution principles for a mobile throughput guidance exposure mechanism that can be used to assist TCP in cellular networks, ensuring high utilization and high service delivery performance.

This document presents a set of considerations and assumptions for the development of a solution. It specifies a protocol that addresses the requirements and the architecture stated in the [Req_Arch_MTG_Exposure]. This document describes also the applicability of the proposed mechanism to video delivery over cellular networks with test results from live production environment.

1.1. Contributing Authors

The editors gratefully acknowledge the following additional contributors: Peter Szilagyi/Nokia, Csaba Vulkan/Nokia, Ram Gopal/Nokia, Guenter Klas/Vodafone and Peter Cosimini/Vodafone.

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 RFC 2119 [RFC2119].

1.3. Acronyms and Abbreviations

This document uses the following acronyms:

ECGI	E-UTRAN Cell Global Identifier format
ECN	Explicit Congestion Notification
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
IV	Initialization Vector
LTE	Long Term Evolution
MTG	Mobile Throughput Guidance
RAN	Radio Access Network
RCTP	RTP Control Protocol
RTT	Round Trip Time
SACK	Selective Acknowledgement
TCP	Transmission Control Protocol
TCP-EDO	TCP Extended Data option
TG	Throughput Guidance
UE	User Equipment

1.4. Definitions

Throughput Guidance Provider:

A functional element that has access to the radio network information and signals to the TCP server, information about the (near-real time) throughput estimated to be available to a UE at the radio downlink interface

1.5. Assumptions and Considerations for the Solution

This document specifies a solution protocol that is complies with the requirements and architecture specified in [Req_Arch_MTG_Exposure]. The protocol is used by the cellular network to provide throughput guidance information to the TCP server; this information indicates the throughput estimated to be available at the radio downlink interface for the TCP connection. The protocol allows the information to be provided in near real time in situations where the network conditions are changing frequently or the user is moving.

While the implementation details can vary according to the access technology, the resource allocation is abstracted as the capacity of the "radio link" between the RAN and the UE. For example, in the case of an LTE network, the number of physical resource blocks allocated to a UE, along with the modulation scheme and coding rate used, can be translated into radio link capacity in Megabits per second (Mbit/s). From the derived UE’s total throughput and with the

UE’s TCP flow information, Throughput guidance for the TCP connection can be computed.

The TCP server can use this explicit information to inform several congestion control decisions. For example: (1) selecting the initial congestion window size, (2) deciding the value of the congestion window during the congestion avoidance phase, and (3) adjusting the size of the congestion window when the conditions on the "radio link" change. In other words, with this additional information, TCP neither has to congest the network when probing for available resources (by increasing its congestion window), nor rely on heuristics to decide how much it should reduce its sending rate after a congestion episode.

The same explicit information can also be used to optimize application behavior given the available resources. For example, when video is encoded in multiple bitrates, the application server can select the highest encoding rate that the network can deliver.

This solution specified in this document also satisfies the following assumptions and considerations:

- The end-to-end traffic is delivered via HTTP.

- The end-to-end traffic is encrypted (through HTTPS), thus HTTP header enrichment cannot be used by intermediate elements between the client and the server.

- TCP is used to deliver the HTTPS traffic.

- The Real-time Transport Protocol (RTP) network protocol is not used for traffic delivery.

The protocol specified in this document assumes that a trustful relationship between the Throughput Guidance Provider and the TCP server has been formed using the means discussed in the Security considerations section.

The solution in this document satisfies the considerations and the assumptions presented above, and proposes an in-band exposure mechanism where the throughput guidance information is added to the TCP headers of the relevant upstream packets. HTTP and TCP are the most prevalent protocols in the Internet, used even by the most popular streaming application. Throughput guidance at TCP level can be shared among multiple applications; it is not limited to any particular application level optimization only but it offers a generic approach that works even if application level end-to-end encryption, such as HTTPS, is applied.

In particular, the Throughput Guidance Providers adds the throughput guidance information to the Options field of the TCP header (see RFC 0793 [RFC0793]) of packets from the TCP client to the TCP server. An in-band mechanism is proposed because it does not require a separate interface, reference value, or correlation mechanism that would be needed with out of band approaches such as with RCTP that is limited to only certain types of applications. Furthermore, an in-band mechanism can keep up with the rapid changes in the underlying radio link throughput. Unlike existing mechanisms such as ECN, where an ECN- aware router sets a mark in the IP header in order to signal impending congestion (see [RFC3168]). The proposed scheme provides explicit information, (termed "Throughput Guidance") about the estimated throughput available for the TCP connection at the radio link between the RAN and the UE.

Note that once standardized and implemented, TCP Extended Data option (TCP-EDO) can be used to carry the throughput guidance information as specified in [tcp-edo] and simplify the use of the TCP Option fields by extending the space available for TCP options. Currently the TCP-EDO is still work in progress and not available in production. Therefore, the use of TCP-EDO to carry throughput guidance is left for the later drafts.

2. Protocol

This section describes the protocol mechanism and the message format that needs to be communicated from the RAN to the TCP remote endpoint. We describe the protocol mechanism and message format for throughput guidance. The protocol mechanism is defined in an extensible way to allow additional information to be specified and communicated. The protocol specification is based on the existing experiments and running code. It is recommended to insert the throughput guidance information to the TCP segments that flow from client to server (see reasoning in "Assumptions and Considerations" section). Most of the time, TCP segments are ACK packets from a client to the server and hence packets are unlikely to be fragmented. However, the described protocol solution can deal with fragmentation.

The Mobile Throughput Guidance Signaling message conveys information on the throughput estimated to be available at the down link path for a given TCP connection. The information is sent to the uplink end-point of the connection (i.e, the TCP server). The TCP server MAY use this information to adapt TCP behavior and to adjust application-level behavior to the link conditions as defined in [Req_Arch_MTG_Exposure].

A good example is a content optimizer or a cache that can adapt the application-level coding to match the indicated downlink radio

conditions. As radio link conditions may change rapidly, this guidance information is best carried in-band using TCP options headers rather than through an out-of-band protocol.

Using the TCP options to carry throughput guidance associates the guidance information with an ongoing TCP connection and explicitly avoids separate session identification information. The proposed mechanism neither impacts the TCP state machine nor the congestion control algorithms of the TCP protocol.

The Options field enables information elements to be inserted into each packet with a 40-byte overall limit; this needs to be shared with the standardized and widely-used option elements, such as the TimeStamp and SACK. (Use of TCP-EDO will lift this constraint once available and deployed). The TCP Options field uses a Kind-Length-Value structure that enables TCP implementations to interpret or ignore information elements in the Options field based on the Kind.

In this draft, we define a message format for encoding information about the estimated capacity of a radio access link between the RAN and the UE which is traversed by a TCP connection. The intention is to define a generic container to convey in-band information within the limited TCP Option space with optional authentication capabilities. This document conveys throughput guidance information. Additional information can be specified in future.

The Throughput Guidance Provider functional element inserts Mobile Throughput Guidance TCP options only if there is enough space in the TCP header. The Throughput Guidance Provider has access to the radio network information and is typically co-located with the RAN functionality.

The Throughput Guidance information must be delivered in a secure way, such than an intermediate node cannot modify it. The information can be provided as plain text in a secure and closed network. In other cases, the information should be authenticated (between the Throughput Guidance Provider and the TCP server). An acceptable level of authentication (according to best common practices) may require more data than fits into a single TCP header (maximum of 40 bytes if no other options are present). As described below, fragmenting information across multiple packets will be used if such is the case.

Two transfer modes are defined to deal with security of exchanged throughput guidance information in this document; namely, plain-text mode and authenticated mode. A third mode, encryption with authentication mode, is equally feasible and may be described in a

future revision of this protocol. The flags field indicate which mode is used.

2.1. Message Format

Mobile Throughput Guidance Signaling uses the common TCP options structure as in [RFC0793] with experimental identifier as defined in [RFC6994]. The followign defines the message format:

```
 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
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |      kind     |    Length     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             ExID              |      Ver      |Resvd|Frag |P|T|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Seq number           |              SBR              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  CL   |KeyIndx|              Auth MAC (20 bytes)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1

Kind:

Code point 253 for Experimental Option for 16-bit ExID [RFC6994]. The size of this field is 1 byte.

Length:

A 1-byte field, length of the option in bytes as defined in RFC793.

ExID:

Two bytes Experimental Identifier according to [RFC6994]. Code point 0x6006.

Ver:

Version of the protocol, set to 1.

Flags:

One byte of MTG protocol flag field as defined below.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|Resvd|Frag |P|T|
+-+-+-+-+-+-+-+-+
```

Flag field of common Kind-Lenght-Value header

Figure 2

Frag:

Three bits that provide information about how to reassemble information if fragmented into multiple packets. If no fragmentation across multiple TCP packet headers is needed, these bits are set to zero. Otherwise, Frag is a counter starting from 1 and incremented by 1 for each subsequent packet of the same type (see P- and T-bits below). For the last fragment, the Fragment is always 7 (binary 111) to indicate that the information is complete.

P and T bits:

These two bits encode the packet type: Plaintext (P=0, T= 0), Cipher text (P=0, T=1), Nonce (IV) (P=1, T=0) or Authentication (P=1, T=1). For Plaintext, the Fragment bits are always zero.

Seq Number:

16-bit sequence number to protect against replay attacks

SBR:

Suggested bit rate for the data session in Mbps. The 12 most significant bits are used for the integer value while the bottom 4 bits correspond to the decimal portion of the throughput value.

CL:

Cell Congestion Level (0, 1, 2, 3). A 4-bit field that indicates the current cell congestion level. "0" indicates no congestion and "3" indicates high congestion value.

Key Index:

A 4-bit field to identify the key used for integrity protection.

Auth MAC:

20 bytes of MAC that protects the TCP option

2.2. Authentication

Authentication covers the entire TCP option, exlcuding the Flags field and the Auth MAC field. The authentication uses HMAC codes (e.g. HMAC- SHA2-224), 128 bits (16 bytes) key size, 256 bits (32 bytes) digest size. Multiple keys (at most 256) for authentication with the same information receiver can be used. Truncation is possible but at least 160 bits (20 bytes) must be used from the digest to meet the typical security level of mobile networks.

Authentication takes a key, the input (arbitrary length) and produces a 32 byte long digest, which is truncated to 20 bytes (keeping the most significant bytes). The HMAC algorithm and truncation can be negotiated via key management (out of scope of this document).

The order in which the fields are included into the message authentication code is the same as the order in which the bytes appear in the message format.

In case the option packets used as input to the HMAC are fragmented into multiple TCP headers, they are processed so that headers with cipher text option are processed first, followed by IV/Nonce option packets.

The options containing the result of the HMAC are marked by setting both P- and T-bits of the flag-field to one. Key Index is set to point to the used authentication key, followed by the resulting authentication code. If the option doesn’t fit into the free option space in the TCP header, it is fragmented across multiple TCP headers in the same way as the cipher text options.

3. Applicability to Video Delivery Optimization

The applicability of the protocol specified in this document to mobile video delivery optimization has been evaluated and tested in different network load scenarios.

In this use case, TCP traffic, for which throughput guidance information is required, passes through a Radio Analytics application which resides in a Mobile-edge Computing (MEC) server (see [MEC_White_Paper]). This Radio Analytics application acts as the Throughput Guidance Provider and sends throughput guidance information for a TCP connection using the Options field in the TCP header (according to the message specification provided in section 2). The TCP server MAY use this information to assist TCP congestion control decisions as described above. The information MAY also be

used to select the application level coding so that it matches the estimated capacity at the radio downlink for that TCP connection.

All of these improvements aim to enhance the quality of experience of the end user by reducing the time-to-start of the content as well as video stall occurrences.

3.1. Test Results

Nokia Networks and Google tested the video delivery optimization use case in a live production LTE network. Google server was placed close to the packet core network of LTE (SGi-interface of LTE). Different network load scenarios were taken into consideration. TCP CUBIC was used in these tests [MTG_ICCRG].

Field trial performance results

Performance metric	Difference of Averages (%)	Diff of 99th percentiles
Time to play	-8.0%	-12%
Number of formats	+4.1%	+29.9%
Client bandwidth	+0.7%	+8.0%
Ave Video resolution	+6.2%	+5.6%
Re-buffer time	-19.7%	-5.1%

Table 1: Performance Data

These user experience improvements results into better video play and are likely to offer longer battery life.

Table 3 summarizes the results from a field trial in the 3G network of a Tier 1 mobile network operator in the Americas. As in the previous case, the Google servers were located close to the packet core network while the network elements from Vasona Networks generated the Throughput Guidance messages.

It is interesting to note that the improvements that Throughput Guidance provides are qualitatively different in LTE and 3G networks. LTE networks generally have capacities that cannot be fully utilized by CUBIC;s initial sending rate. On the other hand, Throughput Guidance tends to be more conservative in 3G networks leading to higher time to play. Video re-buffering is smaller in both cases.

3G field trial preformance results

Performance metric	Difference of Averages (%)	Diff of 99th percentiles
Time to play	+0.8%	+4.6%
Number of formats	-1.2%	+0.0%
Client bandwidth	+2.5%	-0.5%
Ave Video resolution	+0.5%	-1.9%
Re-buffer time	-13.9%	-10.9%

Table 2: Performance Data from 3G field trial

4. Manageability considerations

The application in the RAN SHOULD be configured with a list of destinations to which throughput guidance should be provided. The application in RAN will supply mobile throughput guidance information to more than one TCP server simultaneously based on the list of destinations.

In addition, it SHOULD be possible to configure the frequency (in milliseconds) at which throughput guidance needs to be signaled as well as the required security level and parameters for the encryption and the authentication if supported.

5. Security considerations

Throughput guidance SHOULD be provided in a secure way. The information can be provided as plain text in a secure and closed network (e.g. inside operator network). In other cases, the information should be authenticated (between the Throughput Guidance Provider and the TCP server).

Section 2 described how the TCP Header information is protected. An out-of-band mechanism is currently used to agree upon the set of keys used to authenticate the messages exchanged between the endpoint and the network element that generates the throughput guidance headers. For example, service providers/OTTs can provide a portal that network providers can use to configure the keys they use to encrypt/sign the throughput guidance information. Then, a service behind the portal ensures that the keys are distributed to the servers that need them.

As stated in [Req_Arch_MTG_Exposure], the policy configuration of the Throughput Guidance Provider and the server endpoint, as well as the

key management are beyond the scope of this protocol definition. The protocol assumes that a trustful relationship has been formed between the Throughput Guidance Provider and the TCP server and that the required security level is already configured by the operator and agreed between the entities (i.e. authentication, encryption or both).

The identity of the Mobile Throughput Guidance provider that injects the throughput guidance header must be explicitly known to the endpoint receiving the information. Omitting such information would enable malicious third parties to inject erroneous information.

Fortunately, the issue of malicious disinformation can be easily addressed using well known techniques. First, the network entity responsible for injecting the throughput guidance header can include a cryptographically secure message authentication code. In this way the transport endpoint that receives the throughput guidance header can check that the information was sent by a legitimate entity and that the information has not been tampered with.

Furthermore, the throughput guidance information should be treated only as an estimate to the congestion control algorithm running at the transport endpoint. The endpoint that receives this information should not assume that it is always correct and accurate. Specifically, endpoints should check the validity of the information received and if they find it erroneous they should discard it and possibly take other corrective actions (e.g., discard all future throughput guidance information from a particular IP prefix). Endpoints MUST process throughput guidance information only from TCP segments that would be otherwise be accepted as part of the standard TCP input process. For example, the receiver should ignore throughput guidance information included in TCP ACKs whose acknowledgement sequence numbers fall outside the range of valid sequence numbers.

6. IANA considerations

In the current version of the document and for field tests, the experimental value 253 is used for the "Throughput Guidance" TCP option kind. ExpID SHOULD be set to 0x6006 (16 bits)

7. Acknowledgements

8. References

8.1. Normative References

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

[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>.

[RFC6994] Touch, J., "Shared Use of Experimental TCP Options", RFC 6994, DOI 10.17487/RFC6994, August 2013, <http://www.rfc-editor.org/info/rfc6994>.

8.2. Informative References

[I-D.narten-iana-considerations-rfc2434bis] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", draft-narten-iana-considerations-rfc2434bis-09 (work in progress), March 2008.

[MEC_White_Paper] ETSI, "Mobile-Edge Computing - Introductory Technical White Paper", 2014.

[MTG_ICCRG] Szilagyi, P., and Terzis, A., "Mobile Content Delivery Optimization based on Throughput Guidance", Presentation at ICCRG meeting IETF93 (work in progress), July 2015.

[Req_Arch_MTG_Exposure] Jain, A., , Terzis, A., , Sprecher, N., , Arunachalam, S., , Smith, K., , and G. Klas, "Requirements and reference architecture for Mobile Throughput Guidance Exposure", draft-sprecher-mobile-tg-exposure-req-arch-01.txt (work in progress), February 2015.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[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>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

[RFC4413] West, M. and S. McCann, "TCP/IP Field Behavior", RFC 4413, DOI 10.17487/RFC4413, March 2006, <http://www.rfc-editor.org/info/rfc4413>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[tcp-ao-encrypt] Touch, J., , "A TCP Authentication Option Extension for Payload Encryption", draft-touch-tcp-ao-encrypt-02.txt (work in progress), November 2014.

[tcp-edo] Touch, J., and Eddy, W., "TCP Extended Data Offset Option", draft-ietf-tcpm-tcp-edo-01.txt (work in progress), October 2013.

Appendix A.

Authors' Addresses

Ankur Jain
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Phone: +1-925-526-5879
Email: jankur@google.com

Andreas Terzis
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Phone: +1-650-214-5270
Email: aterzis@google.com

Hannu Flinck
Nokia Networks
Karaportti 13
Espoo
FI

Phone: +358504839522
Email: hannu.flinck@nokia-bell-labs.com

Nurit Sprecher
Nokia Networks
Hod HaSharon
IL

Phone: +97297751229
Email: nurit.sprecher@nokia.com

Swaminathan Arunachalam
Nokia Networks
Irving, TX
US

Phone: +19723303204
Email: swaminathan.arunachalam@nokia.com

Kevin Smith
Vodafone
One Kingdom Street, Paddington Central
London W2 6BY
UK

Phone: +19723303204
Email: kevin.smith@vodafone.com

Vijay Devarapalli
Vasona Networks

Email: vijay@vasonanetworks.com

Roni Bar Yanai
Vasona Networks

Email: rbaryanai@vasonanetworks.com

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

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

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

Abstract

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

This document updates RFCs 4787, 5382, and 5508.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

1.1. Scope

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

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

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

1.2. Terminology

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

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

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

2. TCP Session Tracking

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

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

```
                   +----------------------------+
                   |                            |
                   V                            |
                +------+   Client               |
                |CLOSED|-----SYN------+         |
                +------+              |         |
                    ^                 |         |
                    |TCP_TRANS T.O.   |         |
                    |                 V         |
                +-------+          +-------+    |
                | TRANS |          |  INIT |    |
                +-------+          +-------+    |
                  |    ^               |        |
             data pkt   |               |        |
                  | Server/Client RST  |        |
                  |  TCP_EST T.O.      |        |
                  V    |           Server SYN   |
              +--------------+          |        |
              | ESTABLISHED  |<---------+        |
              +--------------+                   |
               |           |                     |
         Client FIN    Server FIN                |
               |           |                     |
               V           V                     |
        +---------+   +----------+               |
        |  C FIN  |   |  S FIN   |               |
        |   RCV   |   |    RCV   |               |
        +---------+   +----------+               |
            |             |                      |
        Server FIN      Client FIN            TCP_TRANS
            |             |                      T.O.
            V             V                       |
        +----------------------+                  |
        |   C FIN + S FIN RCV  |-----------------+
        +----------------------+
    Legend:
      * Messages sent or received from the server are
        prefixed with "Server".
      * Messages sent or received from the client are
        prefixed with "Client".
      * "C" means "Client-side"
      * "S" means "Server-side".
      * TCP_EST T.O: refers to the established connection
        idle timeout as defined in [RFC5382].
      * TCP_TRANS T.O: refers to the transitory connection
        idle timeout as defined in [RFC5382].
```

Figure 1: Simplified version of the TCP State Machine

2.1. TCP Transitory Connection Idle-Timeout

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

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

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

2.2. TCP RST

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

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

Notes:

* Admittedly, the NAT has to verify whether received TCP RST packets belong to a connection. This verification check is required to avoid off-path attacks.

* If the NAT removes immediately the NAT mapping upon receipt of a TCP RST message, stale connections may be maintained by endpoints if the first RST message is lost between the NAT and the recipient.

3. Port Overlapping Behavior

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

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

The following mechanism MAY be implemented by a NAT:

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

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

4. Address Pooling Paired (APP)

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

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

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

Update: This behavior SHOULD apply also for TCP.

5. Endpoint-Independent Mapping (EIM) Protocol Independence

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

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

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

6. Endpoint-Independent Filtering (EIF) Protocol Independence

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

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

This behavior is aligned with the update in Section 5.

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

7. Endpoint-Independent Filtering (EIF) Mapping Refresh

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

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

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

Update: This behavior SHOULD apply also for TCP.

7.1. Outbound Mapping Refresh and Error Packets

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

8. Port Parity

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

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

1. Implicit dynamic mappings refer to mappings that are created as a side effect of traffic such as an outgoing TCP SYN or outgoing UDP packet. Implicit dynamic mappings usually have a finite lifetime, though this lifetime is generally not known to the client using them.

2. Explicit dynamic mappings refer to mappings that are created as a result, for example, of explicit Port Control Protocol (PCP) MAP and PEER requests. Explicit dynamic mappings have a finite lifetime, and this lifetime is communicated to the client.

9. Port Randomization

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

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

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

10. IP Identification (IP ID)

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

11. ICMP Query Mappings Timeout

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

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

12. Hairpinning Support for ICMP Packets

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

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

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

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

13. IANA Considerations

This document does not require any IANA action.

14. Security Considerations

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

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

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

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

- The updates in Section 7 and Section 11 prevent an illegitimate node to maintain mappings activated in the NAT while these mappings should be cleared.

- Port randomization (Section 9) complicates tracking hosts located behind a NAT.

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

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

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

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

15. References

15.1. Normative References

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

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

[RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008, <http://www.rfc-editor.org/info/rfc5382>.

[RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, DOI 10.17487/RFC5508, April 2009, <http://www.rfc-editor.org/info/rfc5508>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <http://www.rfc-editor.org/info/rfc6056>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <http://www.rfc-editor.org/info/rfc6146>.

[RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field", RFC 6864, DOI 10.17487/RFC6864, February 2013, <http://www.rfc-editor.org/info/rfc6864>.

15.2. Informative References

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

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <http://www.rfc-editor.org/info/rfc3022>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

[RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <http://www.rfc-editor.org/info/rfc6269>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <http://www.rfc-editor.org/info/rfc6887>.

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

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

[RFC7753] Sun, Q., Boucadair, M., Sivakumar, S., Zhou, C., Tsou, T., and S. Perreault, "Port Control Protocol (PCP) Extension for Port-Set Allocation", RFC 7753, DOI 10.17487/RFC7753, February 2016, <http://www.rfc-editor.org/info/rfc7753>.

Acknowledgements

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

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

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

Contributors

The following individual contributed text to the document:

Sarat Kamiset, Insieme Networks, United States

Authors' Addresses

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

Email: repenno@cisco.com

Simon Perreault
Jive Communications
Canada

Email: sperreault@jive.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Senthil Sivakumar
Cisco Systems, Inc.
United States

Email: ssenthil@cisco.com

Kengo Naito
NTT
Tokyo
Japan

Email: k.naito@nttv6.jp

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

G. Fairhurst
University of Aberdeen

April 04, 2016

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

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

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

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

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

Section 4 defines requirements for building a Circuit Breaker.

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

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

- o traffic generated by an application at a time when the provisioned network capacity is being utilised for other purposes;

- o routing changes that cause additional traffic to start using the path monitored by the Circuit Breaker;

- o misconfiguration of a service/network device where the capacity available is insufficient to support the current traffic aggregate;

- o misconfiguration of an admission controller or traffic policer that allows more traffic than expected across the path monitored by the Circuit Breaker.

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

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

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

1.1. Types of Circuit Breaker

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

- o Fast-Trip Circuit Breakers: The relatively short timescale used by this form of circuit breaker is intended to provide protection for network traffic from a single flow or related group of flows.

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

- o Managed Circuit Breakers: Utilize the operations and management functions that might be present in a managed service to implement a circuit breaker.

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

2. Terminology

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

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

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

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

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

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

3.1. Functional Components

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

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

```
 +--------+                                                  +--------+
 |Endpoint|                                                  |Endpoint|
 +--+-----+          >>> circuit breaker traffic >>>         +--+-----+
    |                                                           |
    | +-+  +-+  +---------+  +-+  +-+  +-+  +--------+  +-+  +-+ |
    +-+R+--+R+->+ Ingress +--+R+--+R+--+R+--+ Egress |--+R+--+R+-+
      +++  +-+  +------+--+  +-+  +-+  +-+  +-----+--+  +++  +-+
       |         ^     |                         |      |
       |         |  +--+------+            +------+--+   |
       |         |  | Ingress |            | Egress  |   |
       |         |  | Meter   |            | Meter   |   |
       |         |  +----+----+            +----+----+   |
       |         |       |                      |        |
  +-+  |         |  +----+----+                 |        |  +-+
  |R+--+         |  | Measure +<----------------+        +--+R|
  +++            |  +----+----+      Reported               +++
   |             |       |           Egress                  |
   |             |  +----+----+      Measurement             |
+--+-----+       |  | Trigger +                          +--+-----+
|Endpoint|       |  +----+----+                          |Endpoint|
+--------+       |       |                               +--------+
                 +---<---+
                  Reaction
```

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

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

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

```
 +----------+                 +----------+
 | Ingress  |  +-+  +-+  +-+  | Egress   |
 | Endpoint +->+R+--+R+--+R+--+ Endpoint |
 +--+----+--+  +-+  +-+  +-+  +----+-----+
    ^    |                          |
    | +--+------+             +----+----+
    | | Ingress |             | Egress  |
    | | Meter   |             | Meter   |
    | +----+----+             +----+----+
    |      |                          |
    | +--- +----+                     |
    | | Measure +<-----------------+
    | +----+----+      Reported
    |      |           Egress
    | +----+----+      Measurement
    | | Trigger |
    | +----+----+
    |      |
    +---<--+
    Reaction
```

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

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

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

2. An egress meter (at the receiver or tunnel egress) that records the number/bytes received in each measurement interval. This measures the supported load for the flow or set of flows, and

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

3. A method that communicates the measured values at the ingress and egress to the Circuit Breaker Measurement function. This could use several methods including: Sending return measurement packets (or control messages) from a receiver to a trigger function at the sender; an implementation using Operations, Administration and Management (OAM); or sending an in-band signalling datagram to the trigger function. This could also be implemented purely as a control plane function, e.g., using a software-defined network controller.

4. A measurement function that combines the ingress and egress measurements to assess the present level of network congestion. (For example, the loss rate for each measurement interval could be deduced from calculating the difference between ingress and egress counter values.) Note the method does not require high accuracy for the period of the measurement interval (or therefore the measured value, since isolated and/or infrequent loss events need to be disregarded.)

5. A trigger function that determines whether the measurements indicate persistent excessive congestion. This function defines an appropriate threshold for determining that there is persistent excessive congestion between the ingress and egress. This preferably considers a rate or ratio, rather than an absolute value (e.g., more than 10% loss, but other methods could also be based on the rate of transmission as well as the loss rate). The Circuit Breaker is triggered when the threshold is exceeded in multiple measurement intervals (e.g., 3 successive measurements). Designs need to be robust so that single or spurious events do not trigger a reaction.

6. A reaction that is applied at the Ingress when the Circuit Breaker is triggered. This seeks to automatically remove the traffic causing persistent excessive congestion.

7. A feedback control mechanism that triggers when either the receive or ingress and egress measurements are not available, since this also could indicate a loss of control packets (also a symptom of heavy congestion or inability to control the load).

3.2. Other network topologies

A Circuit Breaker can be deployed in networks with topologies different to that presented in figures 1 and 2. This section describes examples of such usage, and possible places where functions can be implemented.

3.2.1. Use with a multicast control/routing protocol

```
 +----------+                 +--------+  +----------+
 | Ingress  |  +-+  +-+  +-+  | Egress |  |  Egress  |
 | Endpoint +->+R+--+R+--+R+--+ Router |--+ Endpoint +->+
 +----+-----+  +-+  +-+  +-+  +---+--+-+  +----+-----+  |
      ^         ^    ^    ^       |  ^         |        |
      |         |    |    |       |  |         |        |
 +----+----+    + - - - < - - - - + |    +----+----+   | Reported
 | Ingress |      multicast Prune  |    | Egress  |   | Ingress
 | Meter   |                       |    | Meter   |   | Measurement
 +---------+                       |    +----+----+   |
                                   |         |        |
                                   |    +----+----+   |
                                   |    | Measure +<--+
                                   |    +----+----+
                                   |         |
                                   |    +----+----+
                             multicast |    | Trigger |
                             Leave     |    +----+----+
                             Message   |         |
                                      +----<----+
```

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

Figure 3 shows one example of how a multicast Circuit Breaker could be implemented at a pair of multicast endpoints (e.g., to implement a Fast-Trip Circuit Breaker, Section 5.1). The ingress endpoint (the sender that sources the multicast traffic) meters the ingress load, generating an ingress measurement (e.g., recording timestamped packet counts), and sends this measurement to the multicast group together with the traffic it has measured.

Routers along a multicast path forward the multicast traffic (including the ingress measurement) to all active endpoint receivers. Each last hop (egress) router forwards the traffic to one or more egress endpoint(s).

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

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

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

3.2.2. Use with control protocols supporting pre-provisioned capacity

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

3.2.3. Unidirectional Circuit Breakers over Controlled Paths

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

4. Requirements for a Network Transport Circuit Breaker

The requirements for implementing a Circuit Breaker are:

1. There needs to be a communication path for control messages to carry measurement data from the ingress meter and from the egress meter to the point of measurement. (Requirements 16-18 relate to the transmission of control messages.)

2. A CB is REQUIRED to define a measurement period over which the CB Measurement function measures the level of congestion or loss. This method does not have to detect individual packet loss, but MUST have a way to know that packets have been lost/ marked from the traffic flow.

3. An egress meter can also count ECN [RFC3168] congestion marks as a part of measurement of congestion, but in this case, loss MUST also be measured to provide a complete view of the level of congestion. For tunnels, [ID-ietf-tsvwg-tunnel-congestion-feedback] describes a way to measure both loss and ECN-marking; these measurements could be used on a relatively short timescale to drive a congestion control response and/or aggregated over a longer timescale with a higher trigger threshold to drive a CB. Subsequent bullet items in this section discuss the necessity of using a longer timescale and a higher trigger threshold.

4. The measurement period used by a CB Measurement function MUST be longer than the time that current Congestion Control algorithms need to reduce their rate following detection of congestion. This is important because end-to-end Congestion Control algorithms require at least one RTT to notify and adjust the traffic when congestion is experienced, and congestion bottlenecks can share traffic with a diverse range of RTTs. The measurement period is therefore expected to be significantly longer than the RTT experienced by the CB itself.

5. If necessary, a CB MAY combine successive individual meter samples from the ingress and egress to ensure observation of an average measurement over a sufficiently long interval. (Note when meter samples need to be combined, the combination needs to reflect the sum of the individual sample counts divided by the total time/volume over which the samples were measured. Individual samples over different intervals can not be directly combined to generate an average value.)

6. A CB MUST be constructed so that it does not trigger under light or intermittent congestion (see requirements 7-9).

7. A CB is REQUIRED to define a threshold to determine whether the measured congestion is considered excessive.

8. A CB is REQUIRED to define the triggering interval, defining the period over which the trigger uses the collected measurements. CBs need to trigger over a sufficiently long period to avoid additionally penalizing flows with a long path RTT (e.g., many path RTTs).

9. A CB MUST be robust to multiple congestion events. This usually will define a number of measured persistent congestion events per triggering period. For example, a CB MAY combine the results of several measurement periods to determine if the CB is triggered (e.g., it is triggered when persistent excessive congestion is detected in 3 of the measurements within the triggering interval).

10. The normal reaction to a trigger SHOULD disable all traffic that contributed to congestion (otherwise, see requirements 11,12).

11. The reaction MUST be much more severe than that of a Congestion Control algorithm (such as TCP’s congestion control [RFC5681] or TCP-Friendly Rate Control, TFRC [RFC5348]), because the CB reacts to more persistent congestion and operates over longer timescales (i.e., the overload condition will have persisted for a longer time before the CB is triggered).

12. A reaction that results in a reduction SHOULD result in reducing the traffic by at least an order of magnitude. A response that achieves the reduction by terminating flows, rather than randomly dropping packets, will often be more desirable to users of the service. A CB that reduces the rate of a flow, MUST continue to monitor the level of congestion and MUST further react to reduce the rate if the CB is again triggered.

13. The reaction to a triggered CB MUST continue for a period that is at least the triggering interval. Operator intervention will usually be required to restore a flow. If an automated response is needed to reset the trigger, then this needs to not be immediate. The design of an automated reset mechanism needs to be sufficiently conservative that it does not adversely interact with other mechanisms (including other CB algorithms that control traffic over a common path). It SHOULD NOT perform an automated reset when there is evidence of continued congestion.

14. A CB trigger SHOULD be regarded as an abnormal network event. As such, this event SHOULD be logged. The measurements that lead to triggering of the CB SHOULD also be logged.

15. The control communication needs to carry measurements (requirement 1) and, in some uses, also needs to transmit trigger messages to the ingress. This control communication may be in-band or out-of-band. The use of in-band communication is RECOMMENDED when either design would be possible. The preferred CB design is one that triggers when it fails to receive measurement reports that indicate an absence of congestion, in contrast to relying on the successful transmission of a "congested" signal back to the sender. (The feedback signal could itself be lost under congestion).

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

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

16. The control communication MUST be designed to be robust to packet loss. A control message can be lost if there is a failure of the communication path used for the control messages, loss is likely to also be experienced during congestion/ overload. This does not imply that it is desirable to provide reliable delivery (e.g., over TCP), since this can incur additional delay in responding to congestion. Appropriate mechanisms could be to duplicate control messages to provide increased robustness to loss, or/and to regard a lack of control traffic as an indication that excessive congestion could be being experienced [ID-ietf-tsvwg-RFC5405.bis]. If control messages traffic are sent over a shared path, it is RECOMMENDED that this control traffic is prioritized to reduce the probability of loss under congestion. Control traffic also needs to be considered when provisioning a network that uses a Circuit Breaker.

17. There are security requirements for the control communication between endpoints and/or network devices (Section 7). The authenticity of the source and integrity of the control messages (measurements and triggers) MUST be protected from off-path attacks. When there is a risk of on-path attack, a cryptographic authentication mechanism for all control/ measurement messages is RECOMMENDED.

5. Examples of Circuit Breakers

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

5.1. A Fast-Trip Circuit Breaker

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

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

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

5.1.1. A Fast-Trip Circuit Breaker for RTP

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

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

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

1. An RTP Circuit Breaker triggers on reported lack of progress.

2. An RTP Circuit Breaker triggers when no receiver reports messages are received.

3. An RTP Circuit Breaker triggers when the long-term RTP throughput (over many RTTs) exceeds a hard upper limit determined by a method that resembles TCP-Friendly Rate Control (TFRC).

4. An RTP Circuit Breaker includes the notion of Media Usability. This Circuit Breaker is triggered when the quality of the

transported media falls below some required minimum acceptable quality.

5.2. A Slow-trip Circuit Breaker

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

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

5.3. A Managed Circuit Breaker

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

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

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

5.3.1. A Managed Circuit Breaker for SAToP Pseudo-Wires

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

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

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

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

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

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

5.3.2. A Managed Circuit Breaker for Pseudowires (PWs)

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

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

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

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

6. Examples where circuit breakers may not be needed.

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

6.1. CBs over pre-provisioned Capacity

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

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

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

6.2. CBs with tunnels carrying Congestion-Controlled Traffic

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

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

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

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

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

7. Security Considerations

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

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

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

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

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

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

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

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

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

8. IANA Considerations

This document makes no request from IANA.

9. Acknowledgments

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

10. Revision Notes

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

Draft 00

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

Draft 01

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

WG Draft 00

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

WG Draft 01

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

WG Draft 02

Minor fixes for typos. Rewritten security considerations section.

WG Draft 03

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

A clear recommendation of intended scope.

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

WG Draft 04

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

WG Draft 05

Added clarifications requested during AD review.

WG Draft 06

Fixed some remaining typos.

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

WG Draft 07

Additional update following review by Bob Briscoe.

WG Draft 08

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

WG Draft 09

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

WG Draft 10

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

WG Draft 11

Minor corrections after second WGLC.

WG Draft 12

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

WG Draft 13

Fixed a typo.

WG Draft 14

Update after IESG discussion, including:

Reworded introduction. Added definition of ECN.

Requirement

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

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

Reworded the security considerations.

Corrections to wording to improve clarity.

WG Draft 15 (incorporating pending corrections)

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

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

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

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

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

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

11. References

11.1. Normative References

[ID-ietf-tsvwg-RFC5405.bis]
Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines (Work-in-Progress)", 2015.

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

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

11.2. Informative References

[ID-ietf-pals-congcons]
Stein, YJ., Black, D., and B. Briscoe, "Pseudowire Congestion Considerations (Work-in-Progress)", 2015.

[ID-ietf-tsvwg-tunnel-congestion-feedback]
Wei, X., Zhu, L., and L. Dend, "Tunnel Congestion Feedback (Work-in-Progress)", 2015.

[Jacobsen88]
European Telecommunication Standards, Institute (ETSI), "Congestion Avoidance and Control", SIGCOMM Symposium proceedings on Communications architectures and protocols", August 1998.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <http://www.rfc-editor.org/info/rfc1112>.

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

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <http://www.rfc-editor.org/info/rfc2914>.

[RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <http://www.rfc-editor.org/info/rfc3985>.

[RFC4488] Levin, O., "Suppression of Session Initiation Protocol (SIP) REFER Method Implicit Subscription", RFC 4488, DOI 10.17487/RFC4488, May 2006, <http://www.rfc-editor.org/info/rfc4488>.

[RFC4553] Vainshtein, A., Ed. and YJ. Stein, Ed., "Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP)", RFC 4553, DOI 10.17487/RFC4553, June 2006, <http://www.rfc-editor.org/info/rfc4553>.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <http://www.rfc-editor.org/info/rfc4601>.

[RFC5086] Vainshtein, A., Ed., Sasson, I., Metz, E., Frost, T., and P. Pate, "Structure-Aware Time Division Multiplexed (TDM) Circuit Emulation Service over Packet Switched Network (CESoPSN)", RFC 5086, DOI 10.17487/RFC5086, December 2007, <http://www.rfc-editor.org/info/rfc5086>.

[RFC5087] Stein, Y(J)., Shashoua, R., Insler, R., and M. Anavi, "Time Division Multiplexing over IP (TDMoIP)", RFC 5087, DOI 10.17487/RFC5087, December 2007, <http://www.rfc-editor.org/info/rfc5087>.

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <http://www.rfc-editor.org/info/rfc5348>.

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

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

[RTP-CB] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions (draft-ietf-avtcore-rtp-circuit-breakers)", February 2014.

Author's Address

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

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

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

R. Geib, Ed.
Deutsche Telekom
D. Black
Dell EMC

December 15, 2016

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

Abstract

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

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

other contexts benefitting from a defined Diffserv interconnection interface.

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

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

1.1. Related work

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

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

1.2. Applicability Statement

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

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

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

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

1.3. Document Organization

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

2. MPLS and the Short Pipe tunnel model

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

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

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

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

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

3. Relationship to RFC5127

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

3.1. RFC5127 Background

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

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

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

- o The available coding space for carrying Traffic Class information (e.g., Diffserv PHB) in MPLS (and Ethernet) is only 3 bits in size, and is intended for more than just Diffserv purposes (see, e.g., [RFC5129]).

- o The common interconnection DSCPs ought not to use all 8 possible values. This leaves space for future standards, for private bilateral agreements and for local use PHBs and DSCPs.

- o Migrations from one Diffserv code point scheme to a different one is another possible application of otherwise unused DSCPs.

3.2. Differences from RFC5127

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

- o It follows RFC2475 in allowing the DSCPs used within a network to differ from those to exchange traffic with other networks (at network edges), but provides support to restore ingress DSCP values if one of the recommended interconnect DSCPs in this draft is used. This results in DSCP remarking at both network ingress and network egress, and this draft assumes that such remarking at network edges is possible for all interface types.

- o Diffserv-Intercon suggests limiting the number of interconnection PHBs per Treatment Aggregate to the minimum required. As further discussed below, the number of PHBs per Treatment Aggregate is no more than two. When two PHBs are specified for a Diffserv-Intercon treatment aggregate, the expectation is that the provider network supports DSCPs for both PHBs, but uses a single MPLS TC for the Treatment Aggregate that contains the two PHBs.

- o Diffserv-Intercon suggests mapping other PHBs and DSCPs into the interconnection Treatment Aggregates as further discussed below.

- o Diffserv-Intercon treats network control traffic as a special case. Within a provider’s network, the CS6 DSCP is used for local

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

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

4. The Diffserv-Intercon Interconnection Classes

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

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

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

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

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

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

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

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

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

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

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

4.1. Diffserv-Intercon Example

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

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

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

```
    Provider O            Provider W
          |                     |
     +----------+          +----------+
     |  AF21    |          |    CS2   |
     +----------+          +----------+
          V                     V
      +~~~~~~~+             +~~~~~~~+
      |Rtr PrO|             |Rtr PrW|              Rtr:   Router
      +~~~~~~~+             +~~~~~~~+            Pr[L]:   Provider[L]
          |        DiffServ      |
     +----------+          +----------+
     |    AF31  |          |    AF31  |
     +----------+          +----------+
          V        Intercon      V
      +~~~~~~~+                 |
      |RtrPrTI|-----------------+           Router Provider T Ingress
      +~~~~~~~+
          |            Provider T domain
     +-----------------+
     |  MPLS TC 2, AF21 |
     +-----------------+
        |      |    +----------+   +~~~~~~~+
        V      '--->|   AF21   |->-|RtrDstH|    Router Destination Host
    +----------+    +----------+   +~~~~~~~+
    |   AF21   |       Local DSCPs Provider T
    +----------+
        |
     +~~~~~~~+
     |RtrPrTE|                              Router Provider T Egress
     +~~~~~~~+
        |        DiffServ
    +----------+
    |    AF31  |
    +----------+
        |        Intercon
     +~~~~~~~+
     |RtrPrF |                              Router Provider F
     +~~~~~~~+
        |
    +----------+
    |   AF11   |    Provider F
    +----------+
```

Diffserv-Intercon example

Figure 1

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

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

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

4.2. End-to-end PHB and DSCP Transparency

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

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

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

- o With the Short Pipe model, the DCSP likely changes and the PHB might change. This document describes a method to simplify Diffserv network interconnection when a DSCP rewrite can’t be avoided.

4.3. Treatment of Network Control traffic at carrier interconnection interfaces

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

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

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

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

- o classification of traffic that is network control traffic for both domains. This traffic should be classified and marked for the CS6 DSCP.

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

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

5. Acknowledgements

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

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

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

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

8. References

8.1. Normative References

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

[RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, DOI 10.17487/RFC2597, June 1999, <http://www.rfc-editor.org/info/rfc2597>.

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

[RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <http://www.rfc-editor.org/info/rfc3270>.

[RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <http://www.rfc-editor.org/info/rfc5129>.

[RFC5865] Baker, F., Polk, J., and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, DOI 10.17487/RFC5865, May 2010, <http://www.rfc-editor.org/info/rfc5865>.

8.2. Informative References

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

[ID.ietf-idr-sla]
IETF, "Inter-domain SLA Exchange", IETF, http://datatracker.ietf.org/doc/draft-ietf-idr-sla-exchange/, 2013.

[IR.34] GSMA Association, "IR.34 Inter-Service Provider IP Backbone Guidelines Version 7.0", GSMA, GSMA IR.34 http://www.gsma.com/newsroom/wp-content/uploads/2012/03/ir.34.pdf, 2012.

[MEF23.1] MEF, "Implementation Agreement MEF 23.1 Carrier Ethernet Class of Service Phase 2", MEF, MEF23.1 http://metroethernetforum.org/PDF_Documents/technical-specifications/MEF_23.1.pdf, 2012.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <http://www.rfc-editor.org/info/rfc2475>.

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

[RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services", RFC 3662, DOI 10.17487/RFC3662, December 2003, <http://www.rfc-editor.org/info/rfc3662>.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <http://www.rfc-editor.org/info/rfc4594>.

[RFC5127] Chan, K., Babiarz, J., and F. Baker, "Aggregation of Diffserv Service Classes", RFC 5127, DOI 10.17487/RFC5127, February 2008, <http://www.rfc-editor.org/info/rfc5127>.

[RFC5160] Levis, P. and M. Boucadair, "Considerations of Provider-to-Provider Agreements for Internet-Scale Quality of Service (QoS)", RFC 5160, DOI 10.17487/RFC5160, March 2008, <http://www.rfc-editor.org/info/rfc5160>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <http://www.rfc-editor.org/info/rfc7657>.

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

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

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

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

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

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

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

```
        |
       \|/           IPv4, DSCP_send
        V
        |
   Peering Router
        |
       \|/           IPv4, DSCP_send
        V
        |
   MPLS Edge Router
        |          Mark MPLS Label, TC_internal
       \|/         Remark DSCP to
        V            (Inner: IPv4, DSCP_d)
        |
   MPLS Core Router  (penultimate hop label popping)
        |                         \
        |            IPv4, DSCP_d |  The DSCP needs to be in network-
        |                 ^^^^^^^^|  internal Diffserv context. The Core
       \|/                         > Router may require or enforce
        V                         |  that. The Edge Router may wrongly
        |                         |  classify, if the DSCP is not in
        |                        /   network-internal Diffserv context.
   MPLS Edge Router
        |                         \   Traffic leaves the network marked
       \|/           IPv4, DSCP_d |  with the network-internal
        V                         > DSCP_d that must be dealt with
        |                         |  by the next network (downstream).
        |                        /
   Peer Router
        |          Remark DSCP to
       \|/           IPv4, DSCP_send
        V
        |


   Short-Pipe / penultimate hop popping example
```

Figure 2

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

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

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

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

```
   Internal Router
        |
        |   Outer Header
       \|/    IPv4, DSCP_send
        V
        |
   Peering Router
        |  Remark DSCP to
       \|/    IPv4, DSCP_ds-int    Diffserv-Intercon DSCP and PHB
        V
        |
   MPLS Edge Router
        |
        |   Mark  MPLS Label, TC_internal
       \|/  Remark DSCP to
        V     (Inner: IPv4, DSCP_d)   domain internal DSCP for
        |                             the PHB
   MPLS Core Router  (penultimate hop label popping)
        |
        |     IPv4, DSCP_d
        |           ^^^^^^
       \|/
        V
        |
        |
   MPLS Edge Router--------------------+
        |                              |
       \|/  Remark DSCP to            \|/  IPv4, DSCP_d
        V     IPv4, DSCP_ds-int        V
        |                              |
        |                              |
   Peer Router              Domain internal Broadband
        |                        Access Router
       \|/  Remark DSCP to            \|/
        V     IPv4, DSCP_send          V  IPv4, DSCP_d
        |                              |
```

Short-Pipe example with Diffserv-Intercon

Figure 3

Authors' Addresses

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

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

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

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

Transport Area Working Group
Internet-Draft
Updates: 3819 (if approved)
Intended status: Best Current Practice
Expires: 7 June 2024

B. Briscoe
Independent
J. Kaippallimalil
Futurewei

5 December 2023

Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP

draft-ietf-tsvwg-ecn-encap-guidelines-22

Abstract

The purpose of this document is to guide the design of congestion notification in any lower layer or tunnelling protocol that encapsulates IP. The aim is for explicit congestion signals to propagate consistently from lower layer protocols into IP. Then the IP internetwork layer can act as a portability layer to carry congestion notification from non-IP-aware congested nodes up to the transport layer (L4). Following these guidelines should assure interworking among IP layer and lower layer congestion notification mechanisms, whether specified by the IETF or other standards bodies. This document is included in BCP 89 and updates the single paragraph of advice to subnetwork designers about ECN in Section 13 of RFC 3819, by replacing it with a reference to the whole of this document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

ECN is defined in the IP header (IPv4 and IPv6) [RFC3168] to allow a resource to notify the onset of queue build-up without having to drop packets, by explicitly marking a proportion of packets with the congestion experienced (CE) codepoint.

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

* It avoids the delay when recovering from congestion losses, which particularly benefits small flows or real-time flows, making their delivery time predictably short [RFC2884];

* As ECN is used more widely by end-systems, it will gradually remove the need to configure a degree of delay into buffers before they start to notify congestion (the cause of bufferbloat). This is because drop involves a trade-off between sending a timely signal and trying to avoid impairment, whereas ECN is solely a signal not an impairment, so there is no harm triggering it earlier.

Some lower layer technologies (e.g. MPLS, Ethernet) are used to form subnetworks with IP-aware nodes only at the edges. These networks are often sized so that it is rare for interior queues to overflow. However, until recently this was more due to the inability of TCP to saturate the links. For many years, fixes such as window scaling [RFC7323] proved hard to deploy. And the Reno variant of TCP has remained in widespread use despite its inability to scale to high flow rates. However, now that modern operating systems are finally capable of saturating interior links, even the buffers of well-provisioned interior switches will need to signal episodes of queuing.

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

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

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

the outer to the forwarded header. Then the lost signals would never be detected and again congestion would build up further. The guidelines given later require protocol designers to carefully consider incremental deployment, and suggest various safe approaches for different circumstances.

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

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

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

1.1. Update to RFC 3819

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

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

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

1.2. Scope

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

The default ECN semantics are described in [RFC3168] and updated by [RFC8311]. Also, the guidelines for AQM designers [RFC7567] clarify the semantics of both drop and ECN signals from AQM algorithms. [RFC4774] is the appropriate best current practice specification of how algorithms with alternative semantics for the ECN field can be partitioned from Internet traffic that uses the default ECN semantics. There are two main examples for how alternative ECN semantics have been defined in practice:

* RFC 4774 suggests using the ECN field in combination with a Diffserv codepoint such as in PCN [RFC6660], Voice over 3G [UTRAN] or Voice over LTE (VoLTE) [LTE-RA];

* RFC 8311 suggests using the ECT(1) codepoint of the ECN field to indicate alternative semantics such as for the experimental Low Latency Low Loss Scalable throughput (L4S) service [RFC9331]).

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

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

The guidelines in this document do ensure that common encapsulation and decapsulation rules are sufficiently generic to cover cases where ECT(1) is used instead of ECT(0) to identify alternative ECN semantics (as in L4S [RFC9331]) and where ECN marking algorithms use ECT(1) to encode 3 severity levels into the ECN field (e.g. PCN [RFC6660]) rather than the default of 2. All these different semantics for the ECN field work because it has been possible to define common default decapsulation rules that allow for all cases.

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

This document focuses on the congestion notification interface between IP and lower layer or tunnel protocols that can encapsulate IP, where the term 'IP' includes IPv4 or IPv6, unicast, multicast or anycast. However, it is likely that the guidelines will also be useful when a lower layer protocol or tunnel encapsulates itself, e.g. Ethernet MAC in MAC ([IEEE802.1Q]; previously 802.1ah) or when it encapsulates other protocols. In the feed-backward mode, propagation of congestion signals for multicast and anycast packets is out-of-scope (because the complexity would make it unlikely to be attempted).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Further terminology used within this document:

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

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

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

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

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

Outer header: The header added to encapsulate a PDU.

Inner header: The header encapsulated by the outer header.

Outgoing header: The header forwarded by the decapsulator.

CE: Congestion Experienced [RFC3168]

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

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

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

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

Not-ECN-PDU: A PDU at the IP layer or below that is part of a congestion control feedback loop that is not capable of propagating explicit congestion notification signals back to the Load Regulator, because at least one of the nodes necessary to propagate the signals is incapable of doing that propagation. Note that this definition is a property of the feedback-loop, not necessarily of the PDU itself, because in some protocols the PDU will self-describe the property, but in others the property might be carried in a separate control-plane context that is somehow bound to the PDU.

3. Modes of Operation

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

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

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

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

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

3.1. Feed-Forward-and-Up Mode

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

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

This mode is illustrated in Figure 1. Along the middle of the figure, layers 2, 3 and 4 of the protocol stack are shown, and one packet is shown along the bottom as it progresses across the network from source to destination, crossing two subnets connected by a router, and crossing two switches on the path across each subnet. Congestion at the output of the first switch (shown as *) leads to a congestion marking in the L2 header (shown as C in the illustration of the packet). The chevrons show the progress of the resulting congestion indication. It is propagated from link to link across the subnet in the L2 header, then when the router removes the marked L2 header, it propagates the marking up into the L3 (IP) header. The router forwards the marked L3 header into subnet B. The L2 protocol used in subnet B does not support ECN, but the signal proceeds across it in the L3 header.

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

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

```
                        _ _ _
          /_______     |_|_|C|  ACK Packet (V)
          \            |_|_|_|
 +---+        layer: 2 3 4 header                            +---+
 |  <|<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Packet V <<<<<<<<<<<<<|<< |L4
 |   |                            +---+                      | ^ |
 |   | . . . . . . Packet U. . | >>|>>> Packet U >>>>>>>>>>>>|>^ |L3
 |   |     +---+     +---+      | ^ |     +---+     +---+   |   |
 |   |     |  *|>>>>>|>>>|>>>>>|>^ |     |   |     |   |   |   |L2
 |___|_____|___|_____|___|_____|___|_____|___|_____|___|_____|___|
 source          subnet A      router       subnet B         dest
   __ _ _ _|   __ _ _ _|  __ _ _|       __ _ _ _|
  |__|_|_|_|  |__|_|_|C| |__|_|C|      |__|_|C|_|    Data________\
  |__|_|_|_|  |__|_|_|_| |__|_|_|      |__|_|_|_|    Packet (U)  /
 layer:4 3 2A      4 3 2A     4 3           4 3 2B
 header
```

Figure 1: Feed-Forward-and-Up Mode

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

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

3.2. Feed-Up-and-Forward Mode

Ethernet is particularly difficult to extend incrementally to support explicit congestion notification. One way to support ECN in such cases has been to use so called 'layer-3 switches'. These are Ethernet switches that dig into the Ethernet payload to find an IP header and manipulate or act on certain IP fields (specifically Diffserv & ECN). For instance, in Data Center TCP [RFC8257], layer-3 switches are configured to mark the ECN field of the IP header within the Ethernet payload when their output buffer becomes congested. With respect to switching, a layer-3 switch acts solely on the addresses in the Ethernet header; it does not use IP addresses, and it does not decrement the TTL field in the IP header.

```
                    _ _ _
          /_______  | | |C|  ACK packet (V)
          \         |_|_|_|
 +---+        layer: 2 3 4 header                          +---+
 |  <|<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Packet V <<<<<<<<<<<<<|<< |L4
 |   |                           +---+                      | ^ |
 |   | . . .  >>>> Packet U >>>|>>>|>>> Packet U >>>>>>>>>>>>|>^ |L3
 |   |      +--^+     +---+      | v|      +---+     +---+  | ^ |
 |   |      |  *|     |   |      | >|>>>>>|>>>|>>>>>|>>>|>>>>>|>^ |L2
 |___|______|___|_____|___|______|__|______|___|_____|___|_____|___|
 source          subnet E      router       subnet F         dest
   __ _ _ _|   __ _ _ _|  __ _ _|       __ _ _ _|
  | | | | |  | | |C| |  | | |C|       | | |C|C|    Data________\
  |_|_|_|_|  |_|_|_|_|  |_|_|_|       |_|_|_|_|    Packet (U)   /
 layer:4 3 2        4 3 2       4 3            4 3 2
 header
```

Figure 2: Feed-Up-and-Forward Mode

By comparing Figure 2 with Figure 1, it can be seen that subnet E (perhaps a subnet of layer-3 Ethernet switches) works in feed-up-and-forward mode by notifying congestion directly into L3 at the point of congestion, even though the congested switch does not otherwise act at L3. In this example, the technology in subnet F (e.g. MPLS) does support ECN natively, so when the router adds the layer-2 header it copies the ECN marking from L3 to L2 as well, as shown by the 'C's in both layers.

3.3. Feed-Backward Mode

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

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

```
                          _ _ _
          /_______  |_|_|C|  ACK packet (X)
          \         |_|_|_|
 +---+        layer: 2 3 4 header                              +---+
 |  <|<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Packet X <<<<<<<<<<<<<|<<  |L4
 |   |                          +---+                        |  ^  |
 |   |                          |  *|>>> Packet W >>>>>>>>>>>|>^  |L3
 |   |     +---+     +---+      |   |     +---+     +---+    |   |
 |   |     |   |     |   |      |  <|<<<<<|<<<|<(V)<|<<<|    |   |L2
 |___|_____|___|_____|___|_____|___|_____|___|_____|___|_____|___|L2
 source          subnet G      router       subnet H         dest
    __ _ _ _   __ _ _ _   __ _ _        __ _ _ _   later
   |_|_|_|_|  |_|_|_|_|  |_|_|_|       |_|_|C|_|  data________\
   |_|_|_|_|  |_|_|_|_|  |_|_|_|       |_|_|_|_|  packet (W)  /
       4 3 2       4 3 2       4 3           4 3 2

                                   _
                                /__ |C|  Feedback control
                                \   |_|  cell/frame (V)
                                     2
    __ _ _ _   __ _ _ _   __ _ _        __ _ _ _   earlier
   |_|_|_|_|  |_|_|_|_|  |_|_|_|       |_|_|_|_|  data________\
   |_|_|_|_|  |_|_|_|_|  |_|_|_|       |_|_|_|_|  packet (U)  /
layer:  4 3 2       4 3 2       4 3           4 3 2
header
```

Figure 3: Feed-Backward Mode

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

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

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

Quantized congestion notification (QCN [IEEE802.1Q]) would suffer from similar problems if extended to multiple subnets. However, from the start QCN was clearly characterized as solely applicable to a single subnet (see Section 6).

3.4. Null Mode

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

A degenerate example is a point-to-point Ethernet link. Excess loading of the link merely causes the queue from the higher layer to back up, while the lower layer remains immune to congestion. Even a whole meshed subnetwork can be made immune to interior congestion by limiting ingress capacity and sufficient sizing of interior links, e.g. a non-blocking fat-tree network [Leiserson85]. An alternative to fat links near the root is numerous thin links with multi-path routing to ensure even worst-case patterns of load cannot congest any link, e.g. a Clos network [Clos53].

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

Feed-forward-and-up is the mode already used for signalling ECN up the layers through MPLS into IP [RFC5129] and through IP-in-IP tunnels [RFC6040], whether encapsulating with IPv4 [RFC2003], IPv6 [RFC2473] or IPsec [RFC4301]. These RFCs take a consistent approach and the following guidelines are designed to ensure this consistency continues as ECN support is added to other protocols that encapsulate IP. The guidelines are also designed to ensure compliance with the

more general best current practice for the design of alternate ECN schemes given in [RFC4774] and extended by [RFC8311].

The rest of this section is structured as follows:

* Section 4.1 addresses the most straightforward cases, where [RFC6040] can be applied directly to add ECN to tunnels that are effectively IP-in-IP tunnels, but with shim header(s) between the IP headers.

* The subsequent sections give guidelines for adding ECN to a subnet technology that uses feed-forward-and-up mode like IP, but it is not so similar to IP that [RFC6040] rules can be applied directly. Specifically:

  - Sections 4.2, 4.3 and 4.4 respectively address how to add ECN support to the wire protocol and to the encapsulators and decapsulators at the ingress and egress of the subnet.

  - Section 4.5 deals with the special, but common, case of sequences of tunnels or subnets that all use the same technology

  - Section 4.6 deals with the question of reframing when IP packets do not map 1:1 into lower layer frames.

4.1. IP-in-IP Tunnels with Shim Headers

A common pattern for many tunnelling protocols is to encapsulate an inner IP header with shim header(s) then an outer IP header. A shim header is defined as one that is not sufficient alone to forward the packet as an outer header. Another common pattern is for a shim to encapsulate a layer 2 (L2) header, which in turn encapsulates (or might encapsulate) an IP header. [I-D.ietf-tsvwg-rfc6040update-shim] clarifies that RFC 6040 is just as applicable when there are shim(s) and possibly a L2 header between two IP headers.

However, it is not always feasible or necessary to propagate ECN between IP headers when separated by a shim. For instance, it might be too costly to dig to arbitrary depths to find an inner IP header, there may be little or no congestion within the tunnel by design (see null mode in Section 3.4 above), or a legacy implementation might not support ECN. In cases where a tunnel does not support ECN, it is important that the ingress does not copy the ECN field from an inner IP header to an outer. Therefore Section 4 of [I-D.ietf-tsvwg-rfc6040update-shim] requires network operators to configure the ingress of a tunnel that does not support ECN so that it zeros the ECN field in the outer IP header.

Nonetheless, in many cases it is feasible to propagate the ECN field between IP headers separated by shim header(s) and/or a L2 header. Particularly in the typical case when the outer IP header and the shim(s) are added (or removed) as part of the same procedure. Even if the shim(s) encapsulate a L2 header, it is often possible to find an inner IP header within the L2 PDU and propagate ECN between that and the outer IP header. This can be thought of as a special case of the feed-up-and-forward mode (Section 3.2), so the guidelines for this mode apply (Section 5).

Numerous shim protocols have been defined for IP tunnelling. More recent ones e.g. Geneve [RFC8926] and Generic UDP Encapsulation (GUE) [I-D.ietf-intarea-gue] cite and follow RFC 6040. And some earlier ones, e.g. CAPWAP [RFC5415] and LISP [RFC9300], cite RFC 3168, which is compatible with RFC 6040.

However, as Section 9.3 of [RFC3168] pointed out, ECN support needs to be defined for many earlier shim-based tunnelling protocols, e.g. L2TPv2 [RFC2661], L2TPv3 [RFC3931], GRE [RFC2784], PPTP [RFC2637], GTP [GTPv1], [GTPv1-U], [GTPv2-C] and Teredo [RFC4380] as well as some recent ones, e.g. VXLAN [RFC7348], NVGRE [RFC7637] and NSH [RFC8300].

All these IP-based encapsulations can be updated in one shot by simple reference to RFC 6040. However, it would not be appropriate to update all these protocols from within the present guidance document. Instead a companion specification [I-D.ietf-tsvwg-rfc6040update-shim] has been prepared that has the appropriate standards track status to update standards track protocols. For those that are not under IETF change control [I-D.ietf-tsvwg-rfc6040update-shim] can only recommend that the relevant body updates them.

4.2. Wire Protocol Design: Indication of ECN Support

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

A lower layer (or subnet) congestion notification system:

1. SHOULD NOT apply explicit congestion notifications to PDUs that are destined for legacy layer-4 transport implementations that will not understand ECN, and

2. SHOULD NOT apply explicit congestion notifications to PDUs if the egress of the subnet might not propagate congestion notifications onward into the higher layer.

   We use the term ECN-PDUs for a PDU on a feedback loop that will propagate congestion notification properly because it meets both the above criteria. And a Not-ECN-PDU is a PDU on a feedback loop that does not meet at least one of the criteria, and will therefore not propagate congestion notification properly. A corollary of the above is that a lower layer congestion notification protocol:

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

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

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

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

The per-domain checking of ECN support in MPLS [RFC5129] is a good example of a way to avoid sending congestion markings to L4 transports that will not understand them, without using any header space in the subnet protocol.

In MPLS, header space is extremely limited, therefore RFC5129 does not provide a field in the MPLS header to indicate whether the PDU is an ECN-PDU or a Not-ECN-PDU. Instead, interior nodes in a domain are allowed to set explicit congestion indications without checking whether the PDU is destined for a L4 transport that will understand them. Nonetheless, this is made safe by requiring that the network operator upgrades all decapsulating edges of a whole domain at once,

as soon as even one switch within the domain is configured to mark rather than drop some PDUs during congestion. Therefore, any edge node that might decapsulate a packet will be capable of checking whether the higher layer transport is ECN-capable. When decapsulating a CE-marked packet, if the decapsulator discovers that the higher layer (inner header) indicates the transport is not ECN-capable, it drops the packet effectively on behalf of the earlier congested node (see Decapsulation Guideline 1 in Section 4.4).

It was only appropriate to define such an incremental deployment strategy because MPLS is targeted solely at professional operators, who can be expected to ensure that a whole subnetwork is consistently configured. This strategy might not be appropriate for other link technologies targeted at zero-configuration deployment or deployment by the general public (e.g. Ethernet). For such 'plug-and-play' environments it will be necessary to invent a failsafe approach that ensures congestion markings will never fall into black holes, no matter how inconsistently a system is put together. Alternatively, congestion notification relying on correct system configuration could be confined to flavours of Ethernet intended only for professional network operators, such as Provider Backbone Bridges (PBB [IEEE802.1Q]; previously 802.1ah).

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

QCN [IEEE802.1Q] is not intended to extend beyond a single subnet, or to interoperate with ECN. Nonetheless, the way QCN indicates to lower layer devices that the end-points will not understand QCN provides another example that a lower layer protocol designer might be able to mimic for their scenario. An operator can define certain Priority Code Points (PCPs [IEEE802.1Q]; previously 802.1p) to indicate non-QCN frames and an ingress bridge is required to map arriving not-QCN-capable IP packets to one of these non-QCN PCPs.

When drop for non-ECN traffic is deferred to the egress of a subnet, it cannot necessarily be assumed that one ECN mark is equivalent to one drop, as was originally required by [RFC3168]. [RFC8311] updated RFC 3168, to allow experimentation with congestion markings that are not equivalent to drop, in particular for L4S [RFC9331]. ECN support in TRILL [I-D.ietf-trill-ecn-support] is a good example of a way to defer drop to the egress of a subnet both when marks are equivalent to drops (as in RFC 3168) and when they are not (as in L4S). The ECN scheme for MPLS [RFC5129] was defined before L4S, so it only currently supports deferred drop that is equivalent to ECN-marking. Nonetheless, in principle, MPLS (and potentially future L2 protocols) could support L4S marking and copy TRILL's approach for determining the drop level of any non-ECN traffic at the subnet egress.

4.3. Encapsulation Guidelines

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

1. Egress Capability Check: A subnet ingress needs to be sure that the corresponding egress of a subnet will propagate any congestion notification added to the outer header across the subnet. This is necessary in addition to checking that an incoming PDU indicates an ECN-capable (L4) transport. Examples of how this guarantee might be provided include:

   * by configuration (e.g. if any label switches in a domain support ECN marking, [RFC5129] requires all egress nodes to have been configured to propagate ECN)

   * by the ingress explicitly checking that the egress propagates ECN (e.g. an early attempt to add ECN support to TRILL used IS-IS to check path capabilities before adding ECN extension flags to each frame [RFC7780]).

   * by inherent design of the protocol (e.g. by encoding ECN marking on the outer header in such a way that a legacy egress that does not understand ECN will consider the PDU corrupt or invalid and discard it, thus at least propagating a form of congestion signal).

2. Egress Fails Capability Check: If the ingress cannot guarantee that the egress will propagate congestion notification, the ingress SHOULD disable ECN at the lower layer when it forwards

the PDU. An example of how the ingress might disable ECN at the lower layer would be by setting the outer header of the PDU to identify it as a Not-ECN-PDU, assuming the subnet technology supports such a concept.

3. Standard Congestion Monitoring Baseline: Once the ingress to a subnet has established that the egress will correctly propagate ECN, on encapsulation it SHOULD encode the same level of congestion in outer headers as is arriving in incoming headers. For example, it might copy any incoming congestion notification into the outer header of the lower layer protocol.

   This ensures that bulk congestion monitoring of outer headers (e.g. by a network management node monitoring ECN in passing frames) will measure congestion accumulated along the whole upstream path starting from the Load Regulator not just starting from the ingress of the subnet. A node that is not the Load Regulator SHOULD NOT re-initialize the level of CE markings in the outer to zero.

   It would still also be possible to measure congestion introduced across one subnet (or tunnel) by subtracting the level of CE markings on inner headers from that on outer headers (see Appendix C of [RFC6040]). For example:

   * If this guideline has been followed and if the level of CE markings is 0.4% on the outer and 0.1% on the inner, 0.4% congestion has been introduced across all the networks since the load regulator, and 0.3% (= 0.4% - 0.1%) has been introduced since the ingress to the current subnet (or tunnel);

   * Without this guideline, if the subnet ingress had re-initialized the outer congestion level to zero, the outer and inner would measure 0.1% and 0.3%. It would still be possible to infer that the congestion introduced since the Load Regulator was 0.4% (= 0.1% + 0.3%). But only if the monitoring system somehow knows whether the subnet ingress re-initialized the congestion level.

   As long as subnet and tunnel technologies use the standard congestion monitoring baseline in this guideline, monitoring systems will know to use the former approach, rather than having to "somehow know" which approach to use.

4.4. Decapsulation Guidelines

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

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

1. If the arriving inner header is a Not-ECN-PDU it implies the L4 transport will not understand explicit congestion markings. Then:

   * If the outer header carries an explicit congestion marking, it is likely that a protocol error has occurred, so drop is the only indication of congestion that the L4 transport will understand. If the congestion marking is the most severe possible, the packet MUST be dropped. However, if congestion can be marked with multiple levels of severity and the packet's marking is not the most severe, this requirement can be relaxed to: the packet SHOULD be dropped.

   * If the outer is an ECN-PDU that carries no indication of congestion or a Not-ECN-PDU the PDU SHOULD be forwarded, but still as a Not-ECN-PDU.

2. If the outer header does not support explicit congestion notification (a Not-ECN-PDU), but the inner header does (an ECN-PDU), the inner header SHOULD be forwarded unchanged.

3. In some lower layer protocols congestion may be signalled as a numerical level, such as in the control frames of quantized congestion notification (QCN [IEEE802.1Q]). If such a multi-bit encoding encapsulates an ECN-capable IP data packet, a function will be needed to convert the quantized congestion level into the frequency of congestion markings in outgoing IP packets.

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

levels in IP or MPLS [RFC6660] and the default encapsulation and decapsulation rules [RFC6040] are compatible with this interpretation of the ECN field.

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

5. The inner and outer headers might carry a combination of congestion notification fields that should not be possible given any currently used protocol transitions. For instance, if Encapsulation Guideline 3 in Section 4.3 had been followed, it should not be possible to have a less severe indication of congestion in the outer than in the inner. It MAY be appropriate to log unexpected combinations of headers and possibly raise an alarm.

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

4.5. Sequences of Similar Tunnels or Subnets

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

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

Appendix C of RFC6040 describes how a tunnel egress can monitor how much congestion has been introduced within a tunnel. A network operator might want to monitor how much congestion had been introduced within a whole sequence of tunnels. Using the technique in Appendix C of RFC6040 at the final egress, the operator could monitor the whole sequence of tunnels, but only if the above

optimisation were used consistently along the sequence of tunnels, in order to make it appear as a single tunnel. Therefore, tunnel endpoint implementations SHOULD allow the operator to configure whether this optimisation is enabled.

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

4.6. Reframing and Congestion Markings

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

Where an AQM marks the ECN field of IP packets as they queue into a layer-2 link, there will be no problem with framing boundaries, because the ECN markings would be applied directly to IP packets. The guidance in this section is only applicable where an ECN capability is being added to a layer-2 protocol so that layer-2 frames can be ECN-marked by an AQM at layer-2. This would only be necessary where AQM will be applied at pure layer-2 nodes (without IP-awareness).

Where ECN marking has had to be applied at non-IP-aware nodes and framing boundaries do not necessarily align with packet boundaries, the decapsulating IP forwarding node SHOULD propagate ECN markings from layer-2 frame headers to IP packets that may have different boundaries as a consequence of reframing.

Two possible design goals for propagating congestion indications, described in Section 5.3 of [RFC3168] and Section 2.4 of [RFC7141], are:

1. approximate preservation of the presence (and therefore timing) of congestion marks on the L2 frames used to construct an IP packet;

    a. at high frequency of congestion marking, approximate preservation of the proportion of congestion marks arriving and departing;

    b. at low frequency of congestion marking, approximate preservation of the timing of congestion marks arriving and departing.

In either case, an implementation SHOULD ensure that any new incoming congestion indication is propagated immediately, not held awaiting the possibility of further congestion indications to be sufficient to indicate congestion on an outgoing PDU [RFC7141]. Nonetheless, to facilitate pipelined implementation, it would be acceptable for congestion marks to propagate to a slightly later IP packet.

At decapsulation in either case:

* ECN marking propagation logically occurs before application of Decapsulation Guideline 1 in Section 4.4. For instance, if ECN marking propagation would cause an ECN congestion indication to be applied to an IP packet that is a Not-ECN-PDU, then that IP packet is dropped in accordance with Guideline 1;

* where a mix of ECN-PDUs and non-ECN-PDUs arrives to construct the same IP packet, the decapsulation spec SHOULD require that packet to be discarded.

* where a mix of different types of ECN-PDUs arrives to construct the same IP packet, e.g. a mix of frames that map to ECT(0) and ECT(1) IP packets, the decapsulation spec might consider this a protocol error. But, if the lower layer protocol has defined such a mix of types of ECN-PDU as valid, it SHOULD require the resulting IP packet to be set to either ECT(0) or ECT(1). In this case, it SHOULD take into account that the RFC series has so far allowed ECT(0) and ECT(1) to be considered equivalent [RFC3168], or ECT(1) can provide a less severe congestion marking than CE [RFC6040], or ECT(1) can indicate an unmarked but ECN-capable packet that is subject to a different marking algorithm to ECT(0) packets, for example L4S [RFC8311] [RFC9331].

The following are two ways that goal 1 might be achieved, but they are not intended to be the only ways:

* Every IP PDU that is constructed, in whole or in part, from an L2 frame that is marked with a congestion signal, has that signal propagated to it;

* Every L2 frame that is marked with a congestion signal, propagates that signal to one IP PDU which is constructed, in whole or in part, from it. If multiple IP PDUs meet this description, the choice can be made arbitrarily but ought to be consistent.

The following gives one way that goal 2 might be achieved, but it is not intended to be the only way:

* For each of the streams of frames that encapsulate the IP packets of each IP-ECN codepoint and follow the same path through the subnet, a counter ('in') tracks octets arriving within the payload of marked L2 frames and another ('out') tracks octets departing in marked IP packets. While 'in' exceeds 'out', forwarded IP packets are ECN-marked. If 'out' exceeds 'in' for longer than a timeout, both counters are zeroed, to ensure that the start of the next congestion episode propagates immediately. The 'out' counter includes octets in reconstructed IP packets that would have been marked, but had to be dropped because they were Not-ECN-PDUs (by Decapsulation Guideline 1 in Section 4.4).

Generally, the number of L2 frames may be higher (e.g. ATM), similar to, or lower (e.g. 802.11 aggregation at a L2-only station) than the number of IP PDUs, and this distinction may influence the choice of mechanism.

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

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

* are encapsulated in Ethernet headers, which have no support for ECN;

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

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

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

* IPv4 and IPv6 are not the only layer-3 protocols that might be encapsulated by lower layer protocols

* Link-layer encryption might be in use, making the layer-2 payload inaccessible

* Many Ethernet switches do not have 'layer-3 switch' capabilities so they cannot read or modify an IP payload

* It might be costly to find an IP header (IPv4 or IPv6) when it may be encapsulated by more than one lower layer header, e.g. Ethernet MAC in MAC ([IEEE802.1Q]; previously 802.1ah).

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

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

2. The depth of the search for an IP header SHOULD be limited. If an IP header is not found soon enough, or an unrecognized or unreadable header is encountered, the switch SHOULD resort to an alternative means of signalling congestion (e.g. drop, or the native lower layer mechanism if available).

3. It is sufficient to use the first IP header found in the stack; the egress of the relevant tunnel can propagate congestion notification upwards to any more deeply encapsulated IP headers later.

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

It can be seen from Section 3.3 that congestion notification in a subnet using feed-backward mode has generally not been designed to be directly coupled with IP layer congestion notification. The subnet attempts to minimize congestion internally, and if the incoming load at the ingress exceeds the capacity somewhere through the subnet, the layer 3 buffer into the ingress backs up. Thus, a feed-backward mode subnet is in some sense similar to a null mode subnet, in that there is no need for any direct interaction between the subnet and higher layer congestion notification. Therefore no detailed protocol design guidelines are appropriate. Nonetheless, a more general guideline is appropriate:

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

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

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

QCN (also known as backward congestion notification, BCN; see Sections 3033 of [IEEE802.1Q]; previously known as 802.1Qau) uses a feed-backward mode structurally similar to ATM's relative rate mechanism. However, QCN confines its applicability to scenarios such as some data centres where all endpoints are directly attached by the same Ethernet technology. If a QCN subnet were later connected into a wider IP-based internetwork (e.g. when attempting to interconnect multiple data centres) it would suffer the inefficiency shown in Figure 3.

7. IANA Considerations

This section is to be removed before publishing as an RFC.

This memo includes no request to IANA.

8. Security Considerations

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

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

* Congestion exposure (ConEx) for networks to audit that their congestion signals are not being suppressed by other networks or by receivers, and for networks to police that senders are responding sufficiently to the signals, irrespective of the L4 transport protocol used [RFC7713].

* A test for a sender to detect whether a network or the receiver is suppressing congestion signals (for example see 2nd para of Section 20.2 of [RFC3168]).

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

Section 6 gives vulnerability to spoofing as one of the reasons for deprecating feed-backward mode.

9. Conclusions

Following the guidance in this document enables ECN support to be extended consistently to numerous protocols that encapsulate IP (IPv4 and IPv6), so that IP continues to fulfil its role as an end-to-end interoperability layer. This includes:

* A wide range of tunnelling protocols including those with various forms of shim header between two IP headers, possibly also separated by a L2 header;

* A wide range of subnet technologies, particularly those that work in the same 'feed-forward-and-up' mode that is used to support ECN in IP and MPLS.

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

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

10. References

10.1. Normative References

[I-D.ietf-trill-ecn-support]
Eastlake, D. E. and B. Briscoe, "TRILL (TRansparent Interconnection of Lots of Links): ECN (Explicit Congestion Notification) Support", Work in Progress, Internet-Draft, draft-ietf-trill-ecn-support-07, 25 February 2018, <https://datatracker.ietf.org/doc/html/draft-ietf-trill-ecn-support-07>.

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

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

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <https://www.rfc-editor.org/info/rfc3819>.

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

[RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <https://www.rfc-editor.org/info/rfc5129>.

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

[RFC7141] Briscoe, B. and J. Manner, "Byte and Packet Congestion Notification", BCP 41, RFC 7141, DOI 10.17487/RFC7141, February 2014, <https://www.rfc-editor.org/info/rfc7141>.

10.2. Informative References

[ATM-TM-ABR]
Cisco, "Understanding the Available Bit Rate (ABR) Service Category for ATM VCs", Design Technote 10415, 5 June 2005, <https://www.cisco.com/c/en/us/support/docs/asynchronous-transfer-mode-atm/atm-traffic-management/10415-atmabr.html>.

[Buck00] Buckwalter, J.T., "Frame Relay: Technology and Practice", Pub. Addison Wesley ISBN-13: 978-0201485240, 2000.

[Clos53] Clos, C., "A Study of Non-Blocking Switching Networks", Bell Systems Technical Journal 32(2):406424, March 1953.

[GTPv1] 3GPP, "GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface", Technical Specification TS 29.060.

[GTPv1-U] 3GPP, "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", Technical Specification TS 29.281.

[GTPv2-C] 3GPP, "Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C)", Technical Specification TS 29.274.

[I-D.ietf-intarea-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", Work in Progress, Internet-Draft, draft-ietf-intarea-gue-09, 26 October 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-intarea-gue-09>.

[I-D.ietf-tsvwg-rfc6040update-shim]
Briscoe, B., "Propagating Explicit Congestion Notification Across IP Tunnel Headers Separated by a Shim", Work in Progress, Internet-Draft, draft-ietf-tsvwg-rfc6040update-shim-22, 29 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-rfc6040update-shim-22>.

[IEEE802.1Q]
IEEE, "IEEE Standard for Local and Metropolitan Area NetworksVirtual Bridged Local Area NetworksAmendment 6: Provider Backbone Bridges", IEEE Std 802.1Q-2018, July 2018, <https://ieeexplore.ieee.org/document/8403927>.

[ITU-T.I.371]
ITU-T, "Traffic Control and Congestion Control in B-ISDN", ITU-T Rec. I.371 (03/04), March 2004, <https://www.itu.int/rec/T-REC-I.371>.

[Leiserson85]
Leiserson, C.E., "Fat-trees: universal networks for hardware-efficient supercomputing", IEEE Transactions on Computers 34(10):892901, October 1985.

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

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

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

[RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, DOI 10.17487/RFC2637, July 1999, <https://www.rfc-editor.org/info/rfc2637>.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, DOI 10.17487/RFC2661, August 1999, <https://www.rfc-editor.org/info/rfc2661>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <https://www.rfc-editor.org/info/rfc2784>.

[RFC2884] Hadi Salim, J. and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", RFC 2884, DOI 10.17487/RFC2884, July 2000, <https://www.rfc-editor.org/info/rfc2884>.

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

[RFC3931] Lau, J., Ed., Townsley, M., Ed., and I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, DOI 10.17487/RFC3931, March 2005, <https://www.rfc-editor.org/info/rfc3931>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <https://www.rfc-editor.org/info/rfc4380>.

[RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <https://www.rfc-editor.org/info/rfc5415>.

[RFC6633] Gont, F., "Deprecation of ICMP Source Quench Messages", RFC 6633, DOI 10.17487/RFC6633, May 2012, <https://www.rfc-editor.org/info/rfc6633>.

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

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <https://www.rfc-editor.org/info/rfc7323>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/info/rfc7348>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <https://www.rfc-editor.org/info/rfc7567>.

[RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <https://www.rfc-editor.org/info/rfc7637>.

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

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

[RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <https://www.rfc-editor.org/info/rfc8084>.

[RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <https://www.rfc-editor.org/info/rfc8087>.

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

[RFC8257] Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L., and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers", RFC 8257, DOI 10.17487/RFC8257, October 2017, <https://www.rfc-editor.org/info/rfc8257>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

[RFC8311] Black, D., "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation", RFC 8311, DOI 10.17487/RFC8311, January 2018, <https://www.rfc-editor.org/info/rfc8311>.

[RFC8926] Gross, J., Ed., Ganga, I., Ed., and T. Sridhar, Ed., "Geneve: Generic Network Virtualization Encapsulation", RFC 8926, DOI 10.17487/RFC8926, November 2020, <https://www.rfc-editor.org/info/rfc8926>.

[RFC9300] Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos, Ed., "The Locator/ID Separation Protocol (LISP)", RFC 9300, DOI 10.17487/RFC9300, October 2022, <https://www.rfc-editor.org/info/rfc9300>.

[RFC9331] De Schepper, K. and B. Briscoe, Ed., "The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S)", RFC 9331, DOI 10.17487/RFC9331, January 2023, <https://www.rfc-editor.org/info/rfc9331>.

[UTRAN] 3GPP, "UTRAN Overall Description", Technical Specification TS 25.401.

Comments Solicited

This section is to be removed before publishing as an RFC.

Comments and questions are encouraged and very welcome. They can be addressed to the IETF Transport Area working group mailing list <tsvwg@ietf.org>, and/or to the authors.

Acknowledgements

Thanks to Gorry Fairhurst and David Black for extensive reviews. Thanks also to the following reviewers: Joe Touch, Andrew McGregor, Richard Scheffenegger, Ingemar Johansson, Piers O'Hanlon, Donald Eastlake, Jonathan Morton, Markku Kojo, Sebastian Möller, Martin Duke and Michael Welzl, who pointed out that lower layer congestion notification signals may have different semantics to those in IP. Thanks are also due to the tsvwg chairs, TSV ADs and IETF liaison people such as Eric Gray, Dan Romascanu and Gonzalo Camarillo for helping with the liaisons with the IEEE and 3GPP. And thanks to Georg Mayer and particularly to Erik Guttman for the extensive search and categorisation of any 3GPP specifications that cite ECN specifications. Thanks also to the Area Reviewers Dan Harkins, Paul Kyzivat, Sue Hares and Dale Worley.

Bob Briscoe was part-funded by the European Community under its Seventh Framework Programme through the Trilogy project (ICT-216372) for initial drafts then through the Reducing Internet Transport Latency (RITE) project (ICT-317700), and for final drafts (from -18) he was funded by Apple Inc. The views expressed here are solely those of the authors.

Contributors

Pat Thaler
Broadcom Corporation (retired)
CA
USA

Pat was a co-author of this draft, but retired before its publication.

Authors' Addresses

Bob Briscoe
Independent
United Kingdom
Email: ietf@bobbriscoe.net
URI: https://bobbriscoe.net/

John Kaippallimalil
Futurewei
5700 Tennyson Parkway, Suite 600
Plano, Texas 75024
United States of America
Email: kjohn@futurewei.com

Network Working Group
Internet-Draft
Intended status: Standard Track

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

Expires: February 2017 September 30, 2016

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

Abstract

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

Status of This Document

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

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

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

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

Table of Contents

1. Introduction

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

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

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

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

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

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

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

1.1. Terminology

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

Decapsulator: a component performing packet decapsulation at tunnel egress.

ECMP: Equal-Cost Multi-Path.

Encapsulator: a component performing packet encapsulation at tunnel egress.

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

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

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

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

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

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

1.2. Requirements Language

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

2. Applicability Statement

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

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

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

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

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

2.1. GRE-in-UDP Tunnel Requirements

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

2.1.1. Requirements for Default GRE-in-UDP Tunnel

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

1. A UDP checksum SHOULD be used when encapsulating in IPv4.

2. A UDP checksum MUST be used when encapsulating in IPv6.

3. GRE-in-UDP tunnel MUST NOT be deployed or configured to carry traffic that is not congestion controlled. As stated in [RFC5405bis], IP-based unicast traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. A default GRE-in-UDP tunnel is not appropriate for traffic that is not known to be congestion-controlled (e.g., most IP multicast traffic).

4. UDP source port values that are used as a source of flow entropy SHOULD be chosen from the ephemeral port range (49152-65535) [RFC5405bis].

5. The use of the UDP source port MUST be configurable so that a single value can be set for all traffic within the tunnel (this disables use of the UDP source port to provide flow entropy). When a single value is set, a random port SHOULD be selected in order to minimize the vulnerability to off-path attacks [RFC6056].

6. For IPv6 delivery networks, the flow entropy SHOULD also be placed in the flow label field for ECMP per [RFC6438].

7. At the tunnel ingress, any fragmentation of the incoming packet (e.g., because the tunnel has a Maximum Transmission Unit (MTU) that is smaller than the packet) SHOULD be performed before encapsulation. In addition, the tunnel ingress MUST apply the UDP checksum to all encapsulated fragments so that the tunnel egress can validate reassembly of the fragments; it MUST set the same Differentiated Services Code Point (DSCP) value as in the Differentiated Services

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

2.1.2. Requirements for TMCE GRE-in-UDP Tunnel

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

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

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

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

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

3. GRE-in-UDP Encapsulation

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

 IPv4 Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Version|  IHL  |Type of Service|          Total Length         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Identification        |Flags|      Fragment Offset    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Time to Live |Protcol=17(UDP)|          Header Checksum      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       Source IPv4 Address                     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     Destination IPv4 Address                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 UDP Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Source Port = Entropy Value  |  Dest. Port = TBD1/TBD2       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           UDP Length          |        UDP Checksum           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 GRE Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |C| |K|S| Reserved0       | Ver |         Protocol Type         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Checksum (optional)      |       Reserved1 (Optional)    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         Key (optional)                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 Sequence Number (optional)                    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1 UDP+GRE Headers in IPv4

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

 IPv6 Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Version| Traffic Class |           Flow Label                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Payload Length        | NxtHdr=17(UDP)|   Hop Limit   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                                                               +
 |                                                               |
 +                      Outer Source IPv6 Address                +
 |                                                               |
 +                                                               +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                                                               +
 |                                                               |
 +                   Outer Destination IPv6 Address              +
 |                                                               |
 +                                                               +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 UDP Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Source Port = entropy value  |  Dest. Port = TBD1/TBD2       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           UDP Length          |        UDP Checksum           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 GRE Header:
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |C| |K|S| Reserved0       | Ver |         Protocol Type         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Checksum (optional)      |       Reserved1 (Optional)    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        Key (optional)                         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 Sequence Number (optional)                    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2 UDP+GRE Headers in IPv6

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

3.1. IP Header

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

3.2. UDP Header

3.2.1. Source Port

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

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

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

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

3.2.2. Destination Port

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

3.2.3. Checksum

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

3.2.4. Length

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

3.3. GRE Header

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

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

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

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

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

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

4. Encapsulation Process Procedures

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

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

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

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

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

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

4.1. MTU and Fragmentation

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

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

4.2. Differentiated Services and ECN Marking

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

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

5. Use of DTLS

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

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

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

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

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

6. UDP Checksum Handling

6.1. UDP Checksum with IPv4

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

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

6.2. UDP Checksum with IPv6

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

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

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

a. It is known (perhaps through knowledge of equipment types and lower layer checks) that packet corruption is exceptionally unlikely and where the operator is willing to take the risk of undetected packet corruption.

b. It is judged through observational measurements (perhaps of historic or current traffic flows that use a non-zero checksum) that the level of packet corruption is tolerably low and where the operator is willing to take the risk of undetected packet corruption.

c. Carrying applications that are tolerant of mis-delivered or corrupted packets (perhaps through higher layer checksum, validation, and retransmission or transmission redundancy) where the operator is willing to rely on the applications using the tunnel to survive any corrupt packets.

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

a. Use of the UDP checksum with IPv6 MUST be the default configuration of all GRE-in-UDP tunnels.

b. The GRE-in-UDP tunnel implementation MUST comply with all requirements specified in Section 4 of [RFC6936] and with requirement 1 specified in Section 5 of [RFC6936].

c. The tunnel decapsulator SHOULD only allow the use of UDP zero-checksum mode for IPv6 on a single received UDP Destination Port regardless of the encapsulator. The motivation for this requirement is possible corruption of the UDP Destination Port, which may cause packet delivery to the wrong UDP port. If that other UDP port requires the UDP checksum, the mis-delivered packet will be discarded.

d. It is RECOMMENDED that the UDP zero-checksum mode for IPv6 is only enabled for certain selected source addresses. The tunnel decapsulator MUST check that the source and destination IPv6 addresses are valid for the GRE-in-UDP tunnel on which the packet was received if that tunnel uses UDP zero-checksum mode and discard any packet for which this check fails.

e. The tunnel encapsulator SHOULD use different IPv6 addresses for each GRE-in-UDP tunnel that uses UDP zero-checksum mode regardless of the decapsulator in order to strengthen the decapsulator’s check of the IPv6 source address (i.e., the same IPv6 source address SHOULD NOT be used with more than one IPv6 destination address, independent of whether that destination address is a unicast or multicast address). When this is not possible, it is RECOMMENDED to use each source IPv6 address for as few UDP zero-checksum mode GRE-in-UDP tunnels as is feasible.

f. When any middlebox exists on the path of a GRE-in-UDP tunnel, it is RECOMMENDED to use the default mode, i.e. use UDP checksum, to reduce the chance that the encapsulated packets will be dropped.

g. Any middlebox that allows the UDP zero-checksum mode for IPv6 MUST comply with requirement 1 and 8-10 in Section 5 of [RFC6936].

h. Measures SHOULD be taken to prevent IPv6 traffic with zero UDP checksums from "escaping" to the general Internet; see Section 8 for examples of such measures.

i. IPv6 traffic with zero UDP checksums MUST be actively monitored for errors by the network operator. For example, the operator may monitor Ethernet layer packet error rates.

j. If a packet with a non-zero checksum is received, the checksum MUST be verified before accepting the packet. This is regardless of whether the tunnel encapsulator and decapsulator have been configured with UDP zero-checksum mode.

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

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

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

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

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

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

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

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

7. Middlebox Considerations

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

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

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

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

7.1. Middlebox Considerations for Zero Checksums

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

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

8. Congestion Considerations

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

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

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

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

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

- o Physical or logical isolation of the links carrying GRE-in-UDP from the general Internet.

- o Deployment of packet filters that block the UDP ports assigned for GRE-in-UDP.

- o Imposition of restrictions on GRE-in-UDP traffic by software tools used to set up GRE-in-UDP tunnels between specific end systems (as might be used within a single data center) or by tunnel ingress nodes for tunnels that don’t terminate at end systems.

9. Backward Compatibility

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

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

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

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

10. IANA Considerations

IANA is requested to make the following allocations:

One UDP destination port number for the indication of GRE,

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

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

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

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

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

11. Security Considerations

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

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

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

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

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

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

12. Acknowledgements

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

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

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

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

13. Contributors

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

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

Email: david.black@emc.com

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

Email: rcallon@juniper.net

John E. Drake
Juniper Networks

Email: jdrake@juniper.net

Gorry Fairhurst
University of Aberdeen

Email: gorry@erg.abdn.ac.uk

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

Email: fanyb@gsta.com

Adrian Farrel
Juniper Networks

Email: adrian@olddog.co.uk

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

Email: vishwas.manral@hp.com

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

Email: cpignata@cisco.com

14. References

14.1. Normative References

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC1122] Braden, R., "Requirements for Internet Hosts -- Communication Layers", RFC1122, October 1989.

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

[RFC2474] Nichols K., Blake S., Baker F., Black D., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", December 1998.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC2890, September 2000.

[RFC5405bis] Eggert, L., "Unicast UDP Usage Guideline for Application Designers", draft-ietf-tsvwg-rfc5405bis, work in progress.

[RFC6040] Briscoe, B., "Tunneling of Explicit Congestion Notification", RFC6040, November 2010.

[RFC6347] Rescoria, E., Modadugu, N., "Datagram Transport Layer Security Version 1.2", RFC6347, 2012.

[RFC6438] Carpenter, B., Amante, S., "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in tunnels", RFC6438, November, 2011.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, April 2013.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.

14.2. Informative References

[RFC792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC793] DARPA, "Transmission Control Protocol", RFC793, September 1981.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2914] Floyd, S., "Congestion Control Principles", RFC2914, September 2000.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC2983, October 2000.

[RFC4787] Audet, F., et al, "network Address Translation (NAT) Behavioral Requirements for Unicast UDP", RFC4787, January 2007.

[RFC6056] Larsen, M. and Gont, F., "Recommendations for Transport-Protocol Port Randomization", RFC6056, January 2011.

[RFC6438] Carpenter, B., Amante, S., "Using the Ipv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC6438, November 2011.

[RFC7042] Eastlake 3rd, D. and Abley, J., "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameter", RFC7042, October 2013.

[RFC7637] Garg, P. and Wang, Y., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC7637, September 2015.

[RFC7676] Pignataro, C., Bonica, R., Krishnan, S., "IPv6 Support for Generic Routing Encapsulation (GRE)", RFC7676, October 2015.

15. Authors' Addresses

Lucy Yong
Huawei Technologies, USA

Email: lucy.yong@huawei.com

Edward Crabbe
Oracle

Email: edward.crabbe@gmail.com

Xiaohu Xu
Huawei Technologies,
Beijing, China

Email: xuxiaohu@huawei.com

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 April 2022

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

25 October 2021

Stream Control Transmission Protocol (SCTP) Network Address Translation Support

draft-ietf-tsvwg-natsupp-23

Abstract

The Stream Control Transmission Protocol (SCTP) provides a reliable communications channel between two end-hosts in many ways similar to the Transmission Control Protocol (TCP). With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT functions for TCP that allows multiple hosts to reside behind a NAT function and yet share a single IPv4 address, even when two hosts (behind a NAT function) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation (NAPT).

This document describes the protocol extensions needed for the SCTP endpoints and the mechanisms for NAT functions necessary to provide similar features of NAPT in the single point and multipoint traversal scenario.

Finally, a YANG module for SCTP NAT is defined.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Stream Control Transmission Protocol (SCTP) [RFC4960] provides a reliable communications channel between two end-hosts in many ways similar to TCP [RFC0793]. With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT functions for TCP that allows multiple hosts to reside behind a NAT function using private-use addresses (see [RFC6890]) and yet share a single IPv4 address, even when two hosts (behind a NAT function) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation (NAPT). Please note that this document focuses on the case where the NAT function maps a single or multiple internal addresses to a single external address and vice versa.

To date, specialized code for SCTP has not yet been added to most NAT functions so that only a translation of IP addresses is supported. The end result of this is that only one SCTP-capable host can successfully operate behind such a NAT function and this host can only be single-homed. The only alternative for supporting legacy NAT functions is to use UDP encapsulation as specified in [RFC6951].

The NAT function in the document refers to NAPT functions described in Section 2.2 of [RFC3022], NAT64 [RFC6146], or DS-Lite AFTR [RFC6333].

This document specifies procedures allowing a NAT function to support SCTP by providing similar features to those provided by a NAPT for TCP (see [RFC5382] and [RFC7857]), UDP (see [RFC4787] and [RFC7857]), and ICMP (see [RFC5508] and [RFC7857]). This document also specifies a set of data formats for SCTP packets and a set of SCTP endpoint procedures to support NAT traversal. An SCTP implementation supporting these procedures can assure that in both single-homed and multi-homed cases a NAT function will maintain the appropriate state without the NAT function needing to change port numbers.

It is possible and desirable to make these changes for a number of reasons:

* It is desirable for SCTP internal end-hosts on multiple platforms to be able to share a NAT function's external IP address in the same way that a TCP session can use a NAT function.

* If a NAT function does not need to change any data within an SCTP packet, it will reduce the processing burden of NAT'ing SCTP by not needing to execute the CRC32c checksum used by SCTP.

* Not having to touch the IP payload makes the processing of ICMP messages by NAT functions easier.

An SCTP-aware NAT function will need to follow these procedures for generating appropriate SCTP packet formats.

When considering SCTP-aware NAT it is possible to have multiple levels of support. At each level, the Internal Host, Remote Host, and NAT function does or does not support the procedures described in this document. The following table illustrates the results of the various combinations of support and if communications can occur between two endpoints.

Internal Host	NAT Function	Remote Host	Communication
Support	Support	Support	Yes
Support	Support	No Support	Limited
Support	No Support	Support	None
Support	No Support	No Support	None
No Support	Support	Support	Limited
No Support	Support	No Support	Limited
No Support	No Support	Support	None
No Support	No Support	No Support	None

Table 1: Communication possibilities

From the table it can be seen that no communication can occur when a NAT function does not support SCTP-aware NAT. This assumes that the NAT function does not handle SCTP packets at all and all SCTP packets sent from behind a NAT function are discarded by the NAT function. In some cases, where the NAT function supports SCTP-aware NAT, but one of the two hosts does not support the feature, communication can possibly occur in a limited way. For example, only one host can have a connection when a collision case occurs.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

This document uses the following terms, which are depicted in Figure 1. Familiarity with the terminology used in [RFC4960] and [RFC5061] is assumed.

Internal-Address (Int-Addr)
An internal address that is known to the internal host.

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

Internal-VTag (Int-VTag)
The SCTP Verification Tag (VTag) (see Section 3.1 of [RFC4960]) that the internal host has chosen for an association. The VTag is a unique 32-bit tag that accompanies any incoming SCTP packet for this association to the Internal-Address.

Remote-Address (Rem-Addr)
The address that an internal host is attempting to contact.

Remote-Port (Rem-Port)
The port number used by the host holding the Remote-Address.

Remote-VTag (Rem-VTag)
The Verification Tag (VTag) (see Section 3.1 of [RFC4960]) that the host holding the Remote-Address has chosen for an association. The VTag is a unique 32-bit tag that accompanies any outgoing SCTP packet for this association to the Remote-Address.

External-Address (Ext-Addr)
An external address assigned to the NAT function, that it uses as a source address when sending packets towards a Remote-Address.

```
     Internal Network    |         External Network
                         |
               Internal  |   External              Remote
               Address   |   Address  /--\/--\    Address
  +--------+         +-----+        /        \           +--------+
  | Host A |=========| NAT |=======| Network  |==========| Host B |
  +--------+         +-----+        \        /           +--------+
               Internal  |            \--/\--/     Remote
    Internal     Port    |                           Port   Remote
      VTag               |                                   VTag
```

Figure 1: Basic Network Setup

4. Motivation and Overview

4.1. SCTP NAT Traversal Scenarios

This section defines the notion of single and multipoint NAT traversal.

4.1.1. Single Point Traversal

In this case, all packets in the SCTP association go through a single NAT function, as shown in Figure 2.

```
   Internal Network    |       External Network
                       |
                       |               /--\/--\
 +--------+         +-----+           /        \          +--------+
 | Host A |=========| NAT |========= | Network  | ========| Host B |
 +--------+         +-----+           \        /          +--------+
                       |               \--/\--/
                       |
```

Figure 2: Single NAT Function Scenario

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

```
       Internal | External : Internal | External
                |          :          |
                |          :          |       /--\/--\
 +--------+  +-----+       :       +-----+   /        \   +--------+
 | Host A |==| NAT |=======:=======| NAT |==| Network  |==| Host B |
 +--------+  +-----+       :       +-----+   \        /   +--------+
                |          :          |       \--/\--/
                |          :          |
```

Figure 3: Serial NAT Functions Scenario

Although one of the main benefits of SCTP multi-homing is redundant paths, in the single point traversal scenario the NAT function represents a single point of failure in the path of the SCTP multi-homed association. However, the rest of the path can still benefit from path diversity provided by SCTP multi-homing.

The two SCTP endpoints in this case can be either single-homed or multi-homed. However, the important thing is that the NAT function in this case sees all the packets of the SCTP association.

4.1.2. Multipoint Traversal

This case involves multiple NAT functions and each NAT function only sees some of the packets in the SCTP association. An example is shown in Figure 4.

```
             Internal      |      External
                           +------+             /---\/---\
                /=======|NAT A |=========\  /           \
    +--------+ /          +------+          \/            \    +--------+
    | Host A |/             |               |    Network   |===| Host B |
    +--------+\             |               \            /    +--------+
               \           +------+          / \          /
                \=======|NAT B |=========/   \---\/---/
                           +------+
                              |
```

Figure 4: Parallel NAT Functions Scenario

This case does not apply to a single-homed SCTP association (i.e., both endpoints in the association use only one IP address). The advantage here is that the existence of multiple NAT traversal points can preserve the path diversity of a multi-homed association for the entire path. This in turn can improve the robustness of the communication.

4.2. Limitations of Classical NAPT for SCTP

Using classical NAPT possibly results in changing one of the SCTP port numbers during the processing, which requires the recomputation of the transport layer checksum by the NAPT function. Whereas for UDP and TCP this can be done very efficiently, for SCTP the checksum (CRC32c) over the entire packet needs to be recomputed (see Appendix B of [RFC4960] for details of the CRC32c computation). This would considerably add to the NAT computational burden, however hardware support can mitigate this in some implementations.

An SCTP endpoint can have multiple addresses but only has a single port number to use. To make multipoint traversal work, all the NAT functions involved need to recognize the packets they see as belonging to the same SCTP association and perform port number translation in a consistent way. One possible way of doing this is to use a pre-defined table of port numbers and addresses configured within each NAT function. Other mechanisms could make use of NAT to NAT communication. Such mechanisms have not been deployed on a wide scale base and thus are not a preferred solution. Therefore an SCTP variant of NAT function has been developed (see Section 4.3).

4.3. The SCTP-Specific Variant of NAT

In this section it is allowed that there are multiple SCTP capable hosts behind a NAT function that share one External-Address. Furthermore, this section focuses on the single point traversal scenario (see Section 4.1.1).

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

Typically, the NAT function has to maintain a NAT binding table of Internal-VTag, Internal-Port, Remote-VTag, Remote-Port, Internal-Address, and whether the restart procedure is disabled or not. An entry in that NAT binding table is called a NAT-State control block. The function Create() obtains the just mentioned parameters and returns a NAT-State control block. A NAT function MAY allow creating NAT-State control blocks via a management interface.

For SCTP packets coming from the external realm of the NAT function the destination address of the packets has to be replaced with the Internal-Address of the host to which the packet has to be delivered, if a NAT state entry is found. The lookup of the Internal-Address is based on the Remote-VTag, Remote-Port, Internal-VTag and the Internal-Port.

The entries in the NAT binding table need to fulfill some uniqueness conditions. There can not be more than one entry NAT binding table with the same pair of Internal-Port and Remote-Port. This rule can be relaxed, if all NAT binding table entries with the same Internal-Port and Remote-Port have the support for the restart procedure disabled (see Section 5.3.1). In this case there can not be no more than one entry with the same Internal-Port, Remote-Port and Remote-VTag and no more than one NAT binding table entry with the same Internal-Port, Remote-Port, and Int-VTag.

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

```
                                          /--\/--\
   +--------+          +-----+           /        \           +--------+
   | Host A | <------> | NAT | <------> | Network  | <------> | Host B |
   +--------+          +-----+           \        /           +--------+
                                          \--/\---/


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

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

              Translate To:

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

Normally a NAT binding table entry will be created.

However, it is possible that there is already a NAT binding table entry with the same Remote-Port, Internal-Port, and Internal-VTag but different Internal-Address and the restart procedure is disabled. In this case the packet containing the INIT chunk MUST be dropped by the NAT and a packet containing an ABORT chunk SHOULD be sent to the SCTP host that originated the packet with the M bit set and 'VTag and Port Number Collision' error cause (see Section 5.1.1 for the format). The source address of the packet containing the ABORT chunk MUST be the destination address of the packet containing the INIT chunk.

If an outgoing SCTP packet contains an INIT or ASCONF chunk and a matching NAT binding table entry is found, the packet is processed as a normal outgoing packet.

It is also possible that a NAT binding table entry with the same Remote-Port and Internal-Port exists without an Internal-VTag conflict but there exists a NAT binding table entry with the same port numbers but a different Internal-Address and the restart procedure is not disabled. In such a case the packet containing the INIT chunk MUST be dropped by the NAT function and a packet containing an ABORT chunk SHOULD be sent to the SCTP host that originated the packet with the M bit set and 'Port Number Collision' error cause (see Section 5.1.1 for the format).

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

```
                                          /--\/--\
+--------+          +-----+            /        \            +--------+
| Host A | <------> | NAT | <------> | Network  | <------> | Host B |
+--------+          +-----+            \        /            +--------+
                                          \--/\---/

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


                         Translate To:

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

The processing of incoming SCTP packets containing an INIT ACK chunk is illustrated in the following figure. The Lookup() function has as input the Internal-VTag, Internal-Port, Remote-VTag, and Remote-Port. It returns the corresponding entry of the NAT binding table and updates the Remote-VTag by substituting it with the value of the Initiate-Tag of the INIT ACK chunk. The wildcard character signifies that the parameter's value is not considered in the Lookup() function or changed in the Update() function, respectively.

```
                                          /--\/--\
+--------+          +-----+            /        \            +--------+
| Host A | <------> | NAT | <------> | Network  | <------> | Host B |
+--------+          +-----+            \        /            +--------+
                                          \--/\---/

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

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

         Returns(NAT-State control block containing Int-Addr)

               INIT ACK[Initiate-Tag]
 Int-Addr:Int-Port <------ Rem-Addr:Rem-Port
                   Int-VTag
```

In the case where the Lookup function fails because it does not find an entry, the SCTP packet is dropped. If it succeeds, the Update routine inserts the Remote-VTag (the Initiate-Tag of the INIT ACK chunk) in the NAT-State control block.

The processing of incoming SCTP packets containing an ABORT or SHUTDOWN COMPLETE chunk with the T bit set is illustrated in the following figure.

```
                                        /--\/--\
+--------+          +-----+           /        \           +--------+
| Host A | <------> | NAT | <------> | Network  | <------> | Host B |
+--------+          +-----+           \        /           +--------+
                                        \--/\---/

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

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

         Returns(NAT-State control block containing Int-Addr)

 Int-Addr:Int-Port <------ Rem-Addr:Rem-Port
                   Rem-VTag
```

For an incoming packet containing an INIT chunk a table lookup is made only based on the addresses and port numbers. If an entry with a Remote-VTag of zero is found, it is considered a match and the Remote-VTag is updated. If an entry with a non-matching Remote-VTag is found or no entry is found, the incoming packet is silently dropped. If an entry with a matching Remote-VTag is found, the incoming packet is forwarded. This allows the handling of INIT collision through NAT functions.

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

```
                                          /--\/--\
   +--------+          +-----+           /        \            +--------+
   | Host A | <------> | NAT | <------> | Network  | <------> | Host B |
   +--------+          +-----+           \        /            +--------+
                                          \--/\---/

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

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

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

    Int-Addr:Int-Port <------ Rem-Addr:Rem-Port
                      Int-VTag
```

5. Data Formats

This section defines the formats used to support NAT traversal. Section 5.1 and Section 5.2 describe chunks and error causes sent by NAT functions and received by SCTP endpoints. Section 5.3 describes parameters sent by SCTP endpoints and used by NAT functions and SCTP endpoints.

5.1. Modified Chunks

This section presents existing chunks defined in [RFC4960] for which additional flags are specified by this document.

5.1.1. Extended ABORT Chunk

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Type = 6    | Reserved  |M|T|           Length              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   \                                                               \
   /                   zero or more Error Causes                   /
   \                                                               \
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The ABORT chunk is extended to add the new 'M bit'. The M bit indicates to the receiver of the ABORT chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box (e.g., NAT).

[NOTE to RFC-Editor: Assignment of M bit to be confirmed by IANA.]

5.1.2. Extended ERROR Chunk

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 9    | Reserved  |M|T|           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                   zero or more Error Causes                   /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

[NOTE to RFC-Editor: Assignment of M bit to be confirmed by IANA.]

5.2. New Error Causes

This section defines the new error causes added by this document.

5.2.1. VTag and Port Number Collision Error Cause

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Cause Code = 0x00B0        |     Cause Length = Variable   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                             Chunk                             /
/                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the 'VTag and Port Number Collision' Error Cause. IANA is requested to assign the value 0x00B0 for this cause code.

Cause Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

Chunk: variable length

The Cause-Specific Information is filled with the chunk that caused this error. This can be an INIT, INIT ACK, or ASCONF chunk. Note that if the entire chunk will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

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

5.2.2. Missing State Error Cause

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Cause Code = 0x00B1        |     Cause Length = Variable   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                       Original Packet                         /
/                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the 'Missing State' Error Cause. IANA is requested to assign the value 0x00B1 for this cause code.

Cause Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

Original Packet: variable length
The Cause-Specific Information is filled with the IPv4 or IPv6 packet that caused this error. The IPv4 or IPv6 header MUST be included. Note that if the packet will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

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

5.2.3. Port Number Collision Error Cause

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Cause Code = 0x00B2        |     Cause Length = Variable   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                             Chunk                             /
/                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the 'Port Number Collision' Error Cause. IANA is requested to assign the value 0x00B2 for this cause code.

Cause Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

Chunk: variable length
The Cause-Specific Information is filled with the chunk that caused this error. This can be an INIT, INIT ACK, or ASCONF chunk. Note that if the entire chunk will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

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

5.3. New Parameters

This section defines new parameters and their valid appearance defined by this document.

5.3.1. Disable Restart Parameter

This parameter is used to indicate that the restart procedure is requested to be disabled. Both endpoints of an association MUST include this parameter in the INIT chunk and INIT ACK chunk when establishing an association and MUST include it in the ASCONF chunk when adding an address to successfully disable the restart procedure.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Type = 0xC007         |         Length = 4            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Parameter Type: 2 bytes (unsigned integer)

This field holds the IANA defined parameter type for the Disable Restart Parameter. IANA is requested to assign the value 0xC007 for this parameter type.

Parameter Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the parameter. The value MUST be 4.

[NOTE to RFC-Editor: Assignment of parameter type to be confirmed by IANA.]

The Disable Restart Parameter MAY appear in INIT, INIT ACK and ASCONF chunks and MUST NOT appear in any other chunk.

5.3.2. VTags Parameter

This parameter is used to help a NAT function to recover from state loss.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Parameter Type = 0xC008   |     Parameter Length = 16     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ASCONF-Request Correlation ID                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Internal Verification Tag                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Remote Verification Tag                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Parameter Type: 2 bytes (unsigned integer)
This field holds the IANA defined parameter type for the VTags Parameter. IANA is requested to assign the value 0xC008 for this parameter type.

Parameter Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the parameter. The value MUST be 16.

ASCONF-Request Correlation ID: 4 bytes (unsigned integer)
This is an opaque integer assigned by the sender to identify each request parameter. The receiver of the ASCONF Chunk will copy this 32-bit value into the ASCONF Response Correlation ID field of the ASCONF ACK response parameter. The sender of the packet containing the ASCONF chunk can use this same value in the ASCONF ACK chunk to find which request the response is for. The receiver MUST NOT change the value of the ASCONF-Request Correlation ID.

Internal Verification Tag: 4 bytes (unsigned integer)
The Verification Tag that the internal host has chosen for the association. The Verification Tag is a unique 32-bit tag that accompanies any incoming SCTP packet for this association to the Internal-Address.

Remote Verification Tag: 4 bytes (unsigned integer)
The Verification Tag that the host holding the Remote-Address has chosen for the association. The VTag is a unique 32-bit tag that accompanies any outgoing SCTP packet for this association to the Remote-Address.

[NOTE to RFC-Editor: Assignment of parameter type to be confirmed by IANA.]

The VTags Parameter MAY appear in ASCONF chunks and MUST NOT appear in any other chunk.

6. Procedures for SCTP Endpoints and NAT Functions

If an SCTP endpoint is behind an SCTP-aware NAT, a number of problems can arise as it tries to communicate with its peers:

* IP addresses can not be included in the SCTP packet. This is discussed in Section 6.1.

* More than one host behind a NAT function could select the same VTag and source port number when communicating with the same peer server. This creates a situation where the NAT function will not be able to tell the two associations apart. This situation is discussed in Section 6.2.

* If an SCTP endpoint is a server communicating with multiple peers and the peers are behind the same NAT function, then the these peers cannot be distinguished by the server. This case is discussed in Section 6.3.

* A restart of a NAT function during a conversation could cause a loss of its state. This problem and its solution is discussed in Section 6.4.

* NAT functions need to deal with SCTP packets being fragmented at the IP layer. This is discussed in Section 6.5.

* An SCTP endpoint can be behind two NAT functions in parallel providing redundancy. The method to set up this scenario is discussed in Section 6.6.

The mechanisms to solve these problems require additional chunks and parameters, defined in this document, and modified handling procedures from those specified in [RFC4960] as described below.

6.1. Association Setup Considerations for Endpoints

The association setup procedure defined in [RFC4960] allows multi-homed SCTP endpoints to exchange its IP-addresses by using IPv4 or IPv6 address parameters in the INIT and INIT ACK chunks. However, this does not work when NAT functions are present.

Every association setup from a host behind a NAT function MUST NOT use multiple internal addresses. The INIT chunk MUST NOT contain an IPv4 Address parameter, IPv6 Address parameter, or Supported Address Types parameter. The INIT ACK chunk MUST NOT contain any IPv4 Address parameter or IPv6 Address parameter using non-global addresses. The INIT chunk and the INIT ACK chunk MUST NOT contain any Host Name parameters.

If the association is intended to be finally multi-homed, the procedure in Section 6.6 MUST be used.

The INIT and INIT ACK chunk SHOULD contain the Disable Restart parameter defined in Section 5.3.1.

6.2. Handling of Internal Port Number and Verification Tag Collisions

Consider the case where two hosts in the Internal-Address space want to set up an SCTP association with the same service provided by some remote hosts. This means that the Remote-Port is the same. If they both choose the same Internal-Port and Internal-VTag, the NAT function cannot distinguish between incoming packets anymore. However, this is unlikely. The Internal-VTags are chosen at random and if the Internal-Ports are also chosen from the ephemeral port range at random (see [RFC6056]) this gives a 46-bit random number that has to match.

The same can happen with the Remote-VTag when a packet containing an INIT ACK chunk or an ASCONF chunk is processed by the NAT function.

6.2.1. NAT Function Considerations

If the NAT function detects a collision of internal port numbers and verification tags, it SHOULD send a packet containing an ABORT chunk with the M bit set if the collision is triggered by a packet containing an INIT or INIT ACK chunk. If such a collision is triggered by a packet containing an ASCONF chunk, it SHOULD send a packet containing an ERROR chunk with the M bit. The M bit is a new

bit defined by this document to express to SCTP that the source of this packet is a "middle" box, not the peer SCTP endpoint (see Section 5.1.1). If a packet containing an INIT ACK chunk triggers the collision, the corresponding packet containing the ABORT chunk MUST contain the same source and destination address and port numbers as the packet containing the INIT ACK chunk. If a packet containing an INIT chunk or an ASCONF chunk, the source and destination address and port numbers MUST be swapped.

The sender of the packet containing an ERROR or ABORT chunk MUST include the error cause with cause code 'VTag and Port Number Collision' (see Section 5.2.1).

6.2.2. Endpoint Considerations

The sender of the packet containing the INIT chunk or the receiver of a packet containing the INIT ACK chunk, upon reception of a packet containing an ABORT chunk with M bit set and the appropriate error cause code for colliding NAT binding table state is included, SHOULD reinitiate the association setup procedure after choosing a new initiate tag, if the association is in COOKIE-WAIT state. In any other state, the SCTP endpoint MUST NOT respond.

The sender of the packet containing the ASCONF chunk, upon reception of a packet containing an ERROR chunk with M bit set, MUST stop adding the path to the association.

6.3. Handling of Internal Port Number Collisions

When two SCTP hosts are behind an SCTP-aware NAT it is possible that two SCTP hosts in the Internal-Address space will want to set up an SCTP association with the same server running on the same remote host. If the two hosts choose the same internal port, this is considered an internal port number collision.

For the NAT function, appropriate tracking can be performed by assuring that the VTags are unique between the two hosts.

6.3.1. NAT Function Considerations

The NAT function, when processing the packet containing the INIT ACK chunk, SHOULD note in its NAT binding table if the association supports the disable restart extension. This note is used when establishing future associations (i.e. when processing a packet containing an INIT chunk from an internal host) to decide if the connection can be allowed. The NAT function does the following when processing a packet containing an INIT chunk:

* If the packet containing the INIT chunk is originating from an internal port to a remote port for which the NAT function has no matching NAT binding table entry, it MUST allow the packet containing the INIT chunk creating an NAT binding table entry.

* If the packet containing the INIT chunk matches an existing NAT binding table entry, it MUST validate that the disable restart feature is supported and, if it does, allow the packet containing the INIT chunk to be forwarded.

* If the disable restart feature is not supported, the NAT function SHOULD send a packet containing an ABORT chunk with the M bit set.

The 'Port Number Collision' error cause (see Section 5.2.3) MUST be included in the ABORT chunk sent in response to the packet containing an INIT chunk.

If the collision is triggered by a packet containing an ASCONF chunk, a packet containing an ERROR chunk with the 'Port Number Collision' error cause SHOULD be sent in response to the packet containing the ASCONF chunk.

6.3.2. Endpoint Considerations

For the remote SCTP server this means that the Remote-Port and the Remote-Address are the same. If they both have chosen the same Internal-Port the server cannot distinguish between both associations based on the address and port numbers. For the server it looks like the association is being restarted. To overcome this limitation the client sends a Disable Restart parameter in the INIT chunk.

When the server receives this parameter it does the following:

* It MUST include a Disable Restart parameter in the INIT ACK to inform the client that it will support the feature.

* It MUST disable the restart procedures defined in [RFC4960] for this association.

Servers that support this feature will need to be capable of maintaining multiple connections to what appears to be the same peer (behind the NAT function) differentiated only by the VTags.

6.4. Handling of Missing State

6.4.1. NAT Function Considerations

If the NAT function receives a packet from the internal network for which the lookup procedure does not find an entry in the NAT binding table, a packet containing an ERROR chunk SHOULD be sent back with the M bit set. The source address of the packet containing the ERROR chunk MUST be the destination address of the packet received from the internal network. The verification tag is reflected and the T bit is set. Such a packet containing an ERROR chunk SHOULD NOT be sent if the received packet contains an ASCONF chunk with the VTags parameter or an ABORT, SHUTDOWN COMPLETE or INIT ACK chunk. A packet containing an ERROR chunk MUST NOT be sent if the received packet contains an ERROR chunk with the M bit set. In any case, the packet SHOULD NOT be forwarded to the remote address.

If the NAT function receives a packet from the internal network for which it has no NAT binding table entry and the packet contains an ASCONF chunk with the VTags parameter, the NAT function MUST update its NAT binding table according to the verification tags in the VTags parameter and, if present, the Disable Restart parameter.

When sending a packet containing an ERROR chunk, the error cause 'Missing State' (see Section 5.2.2) MUST be included and the M bit of the ERROR chunk MUST be set (see Section 5.1.2).

6.4.2. Endpoint Considerations

Upon reception of this packet containing the ERROR chunk by an SCTP endpoint the receiver takes the following actions:

* It SHOULD validate that the verification tag is reflected by looking at the VTag that would have been included in an outgoing packet. If the validation fails, discard the received packet containing the ERROR chunk.

* It SHOULD validate that the peer of the SCTP association supports the dynamic address extension. If the validation fails, discard the received packet containing the ERROR chunk.

* It SHOULD generate a packet containing a new ASCONF chunk containing the VTags parameter (see Section 5.3.2) and the Disable Restart parameter (see Section 5.3.1) if the association is using the disable restart feature. By processing this packet the NAT function can recover the appropriate state. The procedures for generating an ASCONF chunk can be found in [RFC5061].

The peer SCTP endpoint receiving such a packet containing an ASCONF chunk SHOULD add the address and respond with an acknowledgment if the address is new to the association (following all procedures defined in [RFC5061]). If the address is already part of the association, the SCTP endpoint MUST NOT respond with an error, but instead SHOULD respond with a packet containing an ASCONF ACK chunk acknowledging the address and take no action (since the address is already in the association).

Note that it is possible that upon receiving a packet containing an ASCONF chunk containing the VTags parameter the NAT function will realize that it has an 'Internal Port Number and Verification Tag collision'. In such a case the NAT function SHOULD send a packet containing an ERROR chunk with the error cause code set to 'VTag and Port Number Collision' (see Section 5.2.1).

If an SCTP endpoint receives a packet containing an ERROR chunk with 'Internal Port Number and Verification Tag collision' as the error cause and the packet in the Error Chunk contains an ASCONF with the VTags parameter, careful examination of the association is necessary. The endpoint does the following:

* It MUST validate that the verification tag is reflected by looking at the VTag that would have been included in the outgoing packet. If the validation fails, it MUST discard the packet.

* It MUST validate that the peer of the SCTP association supports the dynamic address extension. If the peer does not support this extension, it MUST discard the received packet containing the ERROR chunk.

* If the association is attempting to add an address (i.e. following the procedures in Section 6.6) then the endpoint MUST NOT consider the address part of the association and SHOULD make no further attempt to add the address (i.e. cancel any ASCONF timers and remove any record of the path), since the NAT function has a VTag collision and the association cannot easily create a new VTag (as it would if the error occurred when sending a packet containing an INIT chunk).

* If the endpoint has no other path, i.e. the procedure was executed due to missing a state in the NAT function, then the endpoint MUST abort the association. This would occur only if the local NAT function restarted and accepted a new association before attempting to repair the missing state (Note that this is no different than what happens to all TCP connections when a NAT function looses its state).

6.5. Handling of Fragmented SCTP Packets by NAT Functions

SCTP minimizes the use of IP-level fragmentation. However, it can happen that using IP-level fragmentation is needed to continue an SCTP association. For example, if the path MTU is reduced and there are still some DATA chunk in flight, which require packets larger than the new path MTU. If IP-level fragmentation can not be used, the SCTP association will be terminated in a non-graceful way. See [RFC8900] for more information about IP fragmentation.

Therefore, a NAT function MUST be able to handle IP-level fragmented SCTP packets. The fragments MAY arrive in any order.

When an SCTP packet can not be forwarded by the NAT function due to MTU issues and the IP header forbids fragmentation, the NAT MUST send back a "Fragmentation needed and DF set" ICMPv4 or PTB ICMPv6 message to the internal host. This allows for a faster recovery from this packet drop.

6.6. Multi Point Traversal Considerations for Endpoints

If a multi-homed SCTP endpoint behind a NAT function connects to a peer, it MUST first set up the association single-homed with only one address causing the first NAT function to populate its state. Then it SHOULD add each IP address using packets containing ASCONF chunks sent via their respective NAT functions. The address used in the Add IP address parameter is the wildcard address (0.0.0.0 or ::0) and the address parameter in the ASCONF chunk SHOULD also contain the VTags parameter and optionally the Disable Restart parameter.

7. SCTP NAT YANG Module

This section defines a YANG module for SCTP NAT.

The terminology for describing YANG data models is defined in [RFC7950]. The meaning of the symbols in tree diagrams is defined in [RFC8340].

7.1. Tree Structure

This module augments NAT YANG module [RFC8512] with SCTP specifics. The module supports both classical SCTP NAT (that is, rewrite port numbers) and SCTP-specific variant where the ports numbers are not altered. The YANG "feature" is used to indicate whether SCTP-specific variant is supported.

The tree structure of the SCTP NAT YANG module is provided below:

```
module: ietf-nat-sctp
  augment /nat:nat/nat:instances/nat:instance
          /nat:policy/nat:timers:
    +--rw sctp-timeout?   uint32
  augment /nat:nat/nat:instances/nat:instance
         /nat:mapping-table/nat:mapping-entry:
    +--rw int-VTag?   uint32 {sctp-nat}?
    +--rw rem-VTag?   uint32 {sctp-nat}?
```

Concretely, the SCTP NAT YANG module augments the NAT YANG module (policy, in particular) with the following:

* The sctp-timeout is used to control the SCTP inactivity timeout. That is, the time an SCTP mapping will stay active without SCTP packets traversing the NAT. This timeout can be set only for SCTP. Hence, "/nat:nat/nat:instances/nat:instance/nat:policy/nat:transport-protocols/nat:protocol-id" MUST be set to '132' (SCTP).

In addition, the SCTP NAT YANG module augments the mapping entry with the following parameters defined in Section 3. These parameters apply only for SCTP NAT mapping entries (i.e., "/nat/instances/instance/mapping-table/mapping-entry/transport-protocol" MUST be set to '132');

* The Internal Verification Tag (Int-VTag)

* The Remote Verification Tag (Rem-VTag)

7.2. YANG Module

```
<CODE BEGINS> file "ietf-nat-sctp@2020-11-02.yang"
module ietf-nat-sctp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nat-sctp";
  prefix nat-sctp;

  import ietf-nat {
    prefix nat;
    reference
      "RFC 8512: A YANG Module for Network Address Translation
                 (NAT) and Network Prefix Translation (NPT)";
  }

  organization
    "IETF TSVWG Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/tsvwg/>
```

```
        WG List:  <mailto:tsvwg@ietf.org>

        Author:  Mohamed Boucadair
                 <mailto:mohamed.boucadair@orange.com>";
     description
       "This module augments NAT YANG module with Stream Control
        Transmission Protocol (SCTP) specifics. The extension supports
        both a classical SCTP NAT (that is, rewrite port numbers)
        and a, SCTP-specific variant where the ports numbers are
        not altered.

        Copyright (c) 2020 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     revision 2019-11-18 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: Stream Control Transmission Protocol (SCTP)
                    Network Address Translation Support";
     }

     feature sctp-nat {
       description
         "This feature means that SCTP-specific variant of NAT
          is supported. That is, avoid rewriting port numbers.";
       reference
         "Section 4.3 of RFC XXXX.";
     }

     augment "/nat:nat/nat:instances/nat:instance"
           + "/nat:policy/nat:timers" {
       when "/nat:nat/nat:instances/nat:instance"
          + "/nat:policy/nat:transport-protocols"
          + "/nat:protocol-id = 132";
       description
         "Extends NAT policy with a timeout for SCTP mapping
          entries.";
```

```
      leaf sctp-timeout {
        type uint32;
        units "seconds";
        description
          "SCTP inactivity timeout. That is, the time an SCTP
           mapping entry will stay active without packets
           traversing the NAT.";
      }
    }

    augment "/nat:nat/nat:instances/nat:instance"
          + "/nat:mapping-table/nat:mapping-entry" {
      when "nat:transport-protocol = 132";
      if-feature "sctp-nat";
      description
        "Extends the mapping entry with SCTP specifics.";

      leaf int-VTag {
         type uint32;
         description
           "The Internal Verification Tag that the internal
            host has chosen for this communication.";
      }
      leaf rem-VTag {
        type uint32;
        description
           "The Remote Verification Tag that the remote
            peer has chosen for this communication.";
      }
    }
  }
  <CODE ENDS>
```

8. Various Examples of NAT Traversals

Please note that this section is informational only.

The addresses being used in the following examples are IPv4 addresses for private-use networks and for documentation as specified in [RFC6890]. However, the method described here is not limited to this NAT44 case.

The NAT binding table entries shown in the following examples do not include the flag indicating whether the restart procedure is supported or not. This flag is not relevant for these examples.

8.1. Single-homed Client to Single-homed Server

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

```
                                          /--\/--\
+--------+          +-----+            /        \           +--------+
| Host A | <------> | NAT | <------> | Network  | <------> | Host B |
+--------+          +-----+            \        /           +--------+
                                          \--/\---/
       +---------+--------+----------+--------+-----------+
NAT    |  Int    |  Int   |   Rem    |   Rem  |    Int    |
       |  VTag   |  Port  |   VTag   |   Port |    Addr   |
       +---------+--------+----------+--------+-----------+

   INIT[Initiate-Tag = 1234]
10.0.0.1:1 ------> 203.0.113.1:2
        Rem-VTtag = 0
```

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

```
       NAT function creates entry:
       +---------+--------+----------+--------+-----------+
NAT    |  Int    |  Int   |   Rem    |   Rem  |    Int    |
       |  VTag   |  Port  |   VTag   |   Port |    Addr   |
       +---------+--------+----------+--------+-----------+
       |  1234   |   1    |    0     |   2    |  10.0.0.1 |
       +---------+--------+----------+--------+-----------+

                                INIT[Initiate-Tag = 1234]
                   192.0.2.1:1 ------------------------> 203.0.113.1:2
                                         Rem-VTtag = 0
```

Host B receives the packet containing an INIT chunk and sends a packet containing an INIT ACK chunk with the NAT's Remote-address as destination address.

```
                                          /--\/--\
   +--------+          +-----+           /        \           +--------+
   | Host A | <------> | NAT | <------> | Network  | <------> | Host B |
   +--------+          +-----+           \        /           +--------+
                                          \--/\---/

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

NAT function updates entry:

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

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

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

```
                                          /--\/--\
   +--------+          +-----+           /        \           +--------+
   | Host A | <------> | NAT | <------> | Network  | <------> | Host B |
   +--------+          +-----+           \        /           +--------+
                                          \--/\---/

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

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


                                        COOKIE ACK
                       192.0.2.1:1 <------------------------ 203.0.113.1:2
                                     Int-VTag = 1234

               COOKIE ACK
   10.0.0.1:1 <------ 203.0.113.1:2
              Int-VTag = 1234
```

8.2. Single-homed Client to Multi-homed Server

The internal client is single-homed whereas the remote server is multi-homed. The client (Host A) sends a packet containing an INIT chunk like in the single-homed case.

```
                                                  +--------+
                                 /--\/--\      /-|Router 1| \
+------+         +-----+      /        \    /  +--------+  \ +------+
| Host | <-----> | NAT | <-> | Network  | ==                =| Host |
|   A  |         +-----+      \        /    \  +--------+  / |   B  |
+------+                       \--/\--/      \-|Router 2|-/  +------+
                                                  +--------+
```

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

```
 INIT[Initiate-Tag = 1234]
10.0.0.1:1 ---> 203.0.113.1:2
       Rem-VTag = 0
```

NAT function creates entry:

```
       +---------+--------+----------+--------+-----------+
NAT    |  Int    |  Int   |   Rem    |   Rem  |    Int    |
       |  VTag   |  Port  |   VTag   |   Port |    Addr   |
       +---------+--------+----------+--------+-----------+
       |  1234   |   1    |    0     |   2    |  10.0.0.1 |
       +---------+--------+----------+--------+-----------+
```

```
                              INIT[Initiate-Tag = 1234]
                 192.0.2.1:1 -------------------------> 203.0.113.1:2
                                   Rem-VTag = 0
```

The server (Host B) includes its two addresses in the INIT ACK chunk.

```
                                                  +--------+
                                  /--\/--\      /-|Router 1| \
   +------+         +-----+      /        \    /  +--------+  \ +------+
   | Host | <-----> | NAT | <-> | Network  | ==                =| Host |
   |   A  |         +-----+      \        /    \  +--------+  / |   B  |
   +------+                       \--/\--/      \-|Router 2|-/  +------+
                                                  +--------+

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

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

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

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

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

```
                                                  +--------+
                                     /--\/--\    /-|Router 1| \
   +------+         +-----+        /        \   /  +--------+  \ +------+
   | Host | <-----> | NAT | <-> | Network  | ==                =| Host |
   |   A  |         +-----+        \        /   \  +--------+  / |   B  |
   +------+                         \--/\--/     \-|Router 2|-/  +------+
                                                  +--------+

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

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


                                      COOKIE ACK
                   192.0.2.1:1 <------------------------- 203.0.113.1:2
                                     Int-VTag = 1234

             COOKIE ACK
   10.0.0.1:1 <--- 203.0.113.1:2
            Int-VTag = 1234
```

8.3. Multihomed Client and Server

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

```
                      +-------+
                   /--| NAT 1 |--\       /--\/--\
        +------+   /   +-------+   \     /        \     +--------+
        | Host |===                 ====| Network  |====| Host B |
        |   A  |   \   +-------+   /     \        /     +--------+
        +------+    \--| NAT 2 |--/       \--/\--/
                      +-------+

               +---------+--------+----------+--------+-----------+
        NAT 1  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
               |  VTag   |  Port  |   VTag   |   Port |    Addr   |
               +---------+--------+----------+--------+-----------+

         INIT[Initiate-Tag = 1234]
        10.0.0.1:1 --------> 203.0.113.1:2
                 Rem-VTag = 0
```

NAT function 1 creates entry:

```
        +---------+--------+----------+--------+-----------+
NAT 1   |  Int    |  Int   |   Rem    |   Rem  |    Int    |
        |  VTag   |  Port  |   VTag   |   Port |    Addr   |
        +---------+--------+----------+--------+-----------+
        |  1234   |    1   |     0    |    2   |  10.0.0.1 |
        +---------+--------+----------+--------+-----------+


                                INIT[Initiate-Tag = 1234]
                      192.0.2.1:1 ---------------------> 203.0.113.1:2
                                          Rem-VTag = 0
```

Host B includes its second address in the INIT ACK.

```
                      +-------+
            /--------| NAT 1 |--------\       /--\/--\
+------+   /         +-------+         \     /        \    +--------+
| Host |===                              ====| Network  |===| Host B |
|   A  |   \         +-------+         /     \        /    +--------+
+------+    \--------| NAT 2 |--------/       \--/\--/
                      +-------+

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

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

```
            +---------+--------+----------+--------+-----------+
     NAT 1  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
            |  VTag   |  Port  |   VTag   |   Port |    Addr   |
            +---------+--------+----------+--------+-----------+
            |  1234   |    1   |    5678  |    2   |  10.0.0.1 |
            +---------+--------+----------+--------+-----------+


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

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

```
                        +-------+
              /--------| NAT 1 |--------\       /--\/--\
   +------+   /         +-------+         \     /        \    +--------+
   | Host |===                              ====| Network  |===| Host B |
   |   A  |   \         +-------+         /     \        /    +--------+
   +------+    \--------| NAT 2 |--------/       \--/\--/
                        +-------+

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

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


                                              COOKIE ACK
                              192.0.2.1:1 <------------------ 203.0.113.1:2
                                            Int-VTag = 1234

              COOKIE ACK
   10.0.0.1:1 <------- 203.0.113.1:2
             Int-VTag = 1234
```

Host A announces its second address in an ASCONF chunk. The address parameter contains a wildcard address (0.0.0.0 or ::0) to indicate that the source address has to be be added. The address parameter within the ASCONF chunk will also contain the pair of VTags (remote and internal) so that the NAT function can populate its NAT binding table entry completely with this single packet.

```
                        +-------+
              /--------| NAT 1 |--------\       /--\/--\
   +------+   /         +-------+         \     /        \    +--------+
   | Host |===                              ====| Network  |===| Host B |
   |   A  |   \         +-------+         /     \        /    +--------+
   +------+    \--------| NAT 2 |--------/       \--/\--/
                        +-------+

   ASCONF [ADD-IP=0.0.0.0, INT-VTag=1234, Rem-VTag = 5678]
   10.1.0.1:1 --------> 203.0.113.129:2
            Rem-VTag = 5678
```

NAT function 2 creates a complete entry:

```
          +---------+--------+----------+--------+-----------+
   NAT 2  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
          |  VTag   |  Port  |   VTag   |   Port |    Addr   |
          +---------+--------+----------+--------+-----------+
          |  1234   |    1   |    5678  |    2   |  10.1.0.1 |
          +---------+--------+----------+--------+-----------+


                     ASCONF [ADD-IP, Int-VTag=1234, Rem-VTag = 5678]
                      192.0.2.129:1 -------------------> 203.0.113.129:2
                                         Rem-VTag = 5678

                                            ASCONF ACK
                      192.0.2.129:1 <------------------- 203.0.113.129:2
                                         Int-VTag = 1234

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

8.4. NAT Function Loses Its State

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

```
                                            /--\/--\
   +--------+              +-----+         /        \         +--------+
   | Host A | <----------> | NAT | <----> | Network  | <----> | Host B |
   +--------+              +-----+         \        /         +--------+
                                            \--/\--/

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

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

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

```
                                          /--\/--\
+--------+              +-----+         /        \          +--------+
| Host A | <----------> | NAT | <----> | Network  | <----> | Host B |
+--------+              +-----+         \        /          +--------+
                                          \--/\--/

  ERROR [M bit, NAT state missing]
10.0.0.1:1 <---------- 203.0.113.1:2
          Rem-VTag = 5678
```

On reception of the packet containing the ERROR chunk, Host A sends a packet containing an ASCONF chunk indicating that the former information has to be deleted and the source address of the actual packet added.

```
                                          /--\/--\
+--------+              +-----+         /        \          +--------+
| Host A | <----------> | NAT | <----> | Network  | <----> | Host B |
+--------+              +-----+         \        /          +--------+
                                          \--/\--/

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

       +---------+--------+----------+--------+-----------+
NAT    |  Int    |  Int   |   Rem    |   Rem  |    Int    |
       |  VTag   |  Port  |   VTag   |   Port |    Addr   |
       +---------+--------+----------+--------+-----------+
       |  1234   |    1   |    5678  |    2   |  10.0.0.1 |
       +---------+--------+----------+--------+-----------+

        ASCONF [ADD-IP, DELETE-IP, Int-VTag=1234, Rem-VTag = 5678]
                       192.0.2.2:1 -----------------> 203.0.113.129:2
                                     Rem-VTag = 5678
```

Host B adds the new source address to this association and deletes all other addresses from this association.

```
                                          /--\/--\
   +--------+              +-----+       /        \         +--------+
   | Host A | <----------> | NAT | <----> | Network  | <----> | Host B |
   +--------+              +-----+       \        /         +--------+
                                          \--/\--/

                                          ASCONF ACK
                         192.0.2.2:1 <----------------- 203.0.113.129:2
                                       Int-VTag = 1234

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

               DATA
   10.0.0.1:1 ----------> 203.0.113.1:2
            Rem-VTag = 5678
                                           DATA
                         192.0.2.2:1 ----------------> 203.0.113.129:2
                                      Rem-VTag = 5678
```

8.5. Peer-to-Peer Communications

If two hosts, each of them behind a NAT function, want to communicate with each other, they have to get knowledge of the peer's external address. This can be achieved with a so-called rendezvous server. Afterwards the destination addresses are external, and the association is set up with the help of the INIT collision. The NAT functions create their entries according to their internal peer's point of view. Therefore, NAT function A's Internal-VTag and Internal-Port are NAT function B's Remote-VTag and Remote-Port, respectively. The naming (internal/remote) of the verification tag in the packet flow is done from the sending host's point of view.

```
             Internal | External           External | Internal
                      |                            |
                      |          /--\/---\         |
   +--------+     +-------+     /         \     +-------+     +--------+
   | Host A |<--->| NAT A |<-->| Network   |<-->| NAT B |<--->| Host B |
   +--------+     +-------+     \         /     +-------+     +--------+
                      |          \--/\---/         |
```

NAT Binding Tables

```
          +---------+--------+----------+--------+-----------+
   NAT A  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
          |  VTag   |  Port  |   VTag   |   Port |    Addr   |
          +---------+--------+----------+--------+-----------+

          +---------+--------+----------+--------+-----------+
   NAT B  |  Int    |  Int   |   Rem    |   Rem  |    Int    |
          |  v-tag  |  port  |   v-tag  |   port |    Addr   |
          +---------+--------+----------+--------+-----------+
```

```
   INIT[Initiate-Tag = 1234]
   10.0.0.1:1 --> 203.0.113.1:2
           Rem-VTag = 0
```

NAT function A creates entry:

```
                +---------+--------+----------+--------+-----------+
        NAT A   |  Int    |  Int   |   Rem    |   Rem  |    Int    |
                |  VTag   |  Port  |   VTag   |   Port |    Addr   |
                +---------+--------+----------+--------+-----------+
                |  1234   |   1    |    0     |   2    |  10.0.0.1 |
                +---------+--------+----------+--------+-----------+
```

```
                          INIT[Initiate-Tag = 1234]
                192.0.2.1:1 ----------------> 203.0.113.1:2
                             Rem-VTag = 0
```

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

```
                +---------+--------+----------+--------+-----------+
        NAT B   |  Int    |  Int   |   Rem    |   Rem  |    Int    |
                |  VTag   |  Port  |   VTag   |   Port |    Addr   |
                +---------+--------+----------+--------+-----------+
```

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

```
          Internal | External           External | Internal
                   |                            |
                   |          /--\/---\         |
+--------+     +-------+     /         \     +-------+     +--------+
| Host A |<--->| NAT A |<-->| Network   |<-->| NAT B |<--->| Host B |
+--------+     +-------+     \         /     +-------+     +--------+
                   |          \--/\---/         |

                                          INIT[Initiate-Tag = 5678]
                                         192.0.2.1:1 <-- 10.1.0.1:2
                                                       Rem-VTag = 0
```

NAT B

Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
5678	2	0	1	10.1.0.1

```
                     INIT[Initiate-Tag = 5678]
          192.0.2.1:1  <--------------- 203.0.113.1:2
                               Rem-VTag = 0
```

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

```
             Internal | External           External | Internal
                      |                            |
                      |          /--\/---\         |
   +--------+     +-------+     /         \     +-------+     +--------+
   | Host A |<--->| NAT A |<-->| Network   |<-->| NAT B |<--->| Host B |
   +--------+     +-------+     \         /     +-------+     +--------+
                      |          \--/\---/         |
```

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

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

```
    INIT[Initiate-tag = 5678]
   10.0.0.1:1 <-- 203.0.113.1:2
             Rem-VTag = 0
```

Host A sends a packet containing an INIT ACK chunk, which can pass through NAT function B:

```
             Internal | External           External | Internal
                      |                            |
                      |          /--\/---\         |
   +--------+     +-------+     /         \     +-------+     +--------+
   | Host A |<--->| NAT A |<-->| Network   |<-->| NAT B |<--->| Host B |
   +--------+     +-------+     \         /     +-------+     +--------+
                      |          \--/\---/         |

   INIT ACK[Initiate-Tag = 1234]
   10.0.0.1:1 --> 203.0.113.1:2
         Rem-VTag = 5678


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

                                       NAT function B updates entry:

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

                                          INIT ACK[Initiate-Tag = 1234]
                                             192.0.2.1:1 --> 10.1.0.1:2
                                                        Rem-VTag = 5678
```

The lookup for COOKIE ECHO and COOKIE ACK is successful.

```
             Internal | External           External | Internal
                      |                            |
                      |          /--\/---\         |
   +--------+     +-------+     /         \     +-------+     +--------+
   | Host A |<--->| NAT A |<-->| Network   |<-->| NAT B |<--->| Host B |
   +--------+     +-------+     \         /     +-------+     +--------+
                      |          \--/\---/         |

                                                     COOKIE ECHO
                                            192.0.2.1:1 <-- 10.1.0.1:2
                                                 Rem-VTag = 1234

                                 COOKIE ECHO
                 192.0.2.1:1 <------------- 203.0.113.1:2
                                 Rem-VTag = 1234

          COOKIE ECHO
   10.0.0.1:1 <-- 203.0.113.1:2
          Rem-VTag = 1234

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

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

                                                       COOKIE ACK
                                            192.0.2.1:1 --> 10.1.0.1:2
                                                  Rem-VTag = 5678
```

9. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to control NAT friendliness.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is extended by supporting one new read/write socket option.

9.1. Get or Set the NAT Friendliness (SCTP_NAT_FRIENDLY)

This socket option uses the option_level IPPROTO_SCTP and the option_name SCTP_NAT_FRIENDLY. It can be used to enable/disable the NAT friendliness for future associations and retrieve the value for future and specific ones.

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id
: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application can fill in an association identifier or SCTP_FUTURE_ASSOC for this query. It is an error to use SCTP_{CURRENT|ALL}_ASSOC in assoc_id.

assoc_value
: A non-zero value indicates a NAT-friendly mode.

10. IANA Considerations

[NOTE to RFC-Editor: "RFCXXXX" is to be replaced by the RFC number you assign this document.]

[NOTE to RFC-Editor: The requested values for the chunk type and the chunk parameter types are tentative and to be confirmed by IANA.]

This document (RFCXXXX) is the reference for all registrations described in this section. The requested changes are described below.

10.1. New Chunk Flags for Two Existing Chunk Types

As defined in [RFC6096] two chunk flags have to be assigned by IANA for the ERROR chunk. The requested value for the T bit is 0x01 and for the M bit is 0x02.

This requires an update of the "ERROR Chunk Flags" registry for SCTP:

ERROR Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	T bit	[RFCXXXX]
0x02	M bit	[RFCXXXX]
0x04	Unassigned	
0x08	Unassigned	
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

Table 2

As defined in [RFC6096] one chunk flag has to be assigned by IANA for the ABORT chunk. The requested value of the M bit is 0x02.

This requires an update of the "ABORT Chunk Flags" registry for SCTP:

ABORT Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	T bit	[RFC4960]
0x02	M bit	[RFCXXXX]
0x04	Unassigned	
0x08	Unassigned	
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

Table 3

10.2. Three New Error Causes

Three error causes have to be assigned by IANA. It is requested to use the values given below.

This requires three additional lines in the "Error Cause Codes" registry for SCTP:

Error Cause Codes

Value	Cause Code	Reference
176	VTag and Port Number Collision	[RFCXXXX]
177	Missing State	[RFCXXXX]
178	Port Number Collision	[RFCXXXX]

Table 4

10.3. Two New Chunk Parameter Types

Two chunk parameter types have to be assigned by IANA. IANA is requested to assign these values from the pool of parameters with the upper two bits set to '11' and to use the values given below.

This requires two additional lines in the "Chunk Parameter Types" registry for SCTP:

Chunk Parameter Types

ID Value	Chunk Parameter Type	Reference
49159	Disable Restart (0xC007)	[RFCXXXX]
49160	VTags (0xC008)	[RFCXXXX]

Table 5

10.4. One New URI

An URI in the "ns" subregistry within the "IETF XML" registry has to be assigned by IANA ([RFC3688]):

URI: urn:ietf:params:xml:ns:yang:ietf-nat-sctp
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

10.5. One New YANG Module

An YANG module in the "YANG Module Names" subregistry within the "YANG Parameters" registry has to be assigned by IANA ([RFC6020]):

Name: ietf-nat-sctp
Namespace: urn:ietf:params:xml:ns:yang:ietf-nat-sctp
Maintained by IANA: N
Prefix: nat-sctp
Reference: RFCXXXX

11. Security Considerations

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

Generic issues related to address sharing are discussed in [RFC6269] and apply to SCTP as well.

For SCTP endpoints not disabling the restart procedure, this document does not add any additional security considerations to the ones given in [RFC4960], [RFC4895], and [RFC5061].

SCTP endpoints disabling the restart procedure, need to monitor the status of all associations to mitigate resource exhaustion attacks by establishing a lot of associations sharing the same IP addresses and port numbers.

In any case, SCTP is protected by the verification tags and the usage of [RFC4895] against off-path attackers.

For IP-level fragmentation and reassembly related issues see [RFC4963].

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All data nodes defined in the YANG module that can be created, modified, and deleted (i.e., config true, which is the default) are considered sensitive. Write operations (e.g., edit-config) applied to these data nodes without proper protection can negatively affect network operations. An attacker who is able to access the SCTP NAT function can undertake various attacks, such as:

* Setting a low timeout for SCTP mapping entries to cause failures to deliver incoming SCTP packets.

* Instantiating mapping entries to cause NAT collision.

12. Normative References

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, <https://www.rfc-editor.org/info/rfc4895>.

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

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <https://www.rfc-editor.org/info/rfc5061>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, DOI 10.17487/RFC6096, January 2011, <https://www.rfc-editor.org/info/rfc6096>.

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

[RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <https://www.rfc-editor.org/info/rfc8512>.

13. Informative References

[DOI_10.1145_1496091.1496095]
Hayes, D., But, J., and G. Armitage, "Issues with network address translation for SCTP", ACM SIGCOMM Computer Communication Review Vol. 39, pp. 23-33, DOI 10.1145/1496091.1496095, December 2008, <https://doi.org/10.1145/1496091.1496095>.

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

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <https://www.rfc-editor.org/info/rfc3022>.

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

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <https://www.rfc-editor.org/info/rfc4963>.

[RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008, <https://www.rfc-editor.org/info/rfc5382>.

[RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, DOI 10.17487/RFC5508, April 2009, <https://www.rfc-editor.org/info/rfc5508>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <https://www.rfc-editor.org/info/rfc6056>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <https://www.rfc-editor.org/info/rfc6269>.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <https://www.rfc-editor.org/info/rfc6333>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <https://www.rfc-editor.org/info/rfc6458>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <https://www.rfc-editor.org/info/rfc6951>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <https://www.rfc-editor.org/info/rfc8900>.

Acknowledgments

The authors wish to thank Mohamed Boucadair, Gorry Fairhurst, Bryan Ford, David Hayes, Alfred Hines, Karen E. E. Nielsen, Henning Peters, Maksim Proshin, Timo Völker, Dan Wing, and Qiaobing Xie for their invaluable comments.

In addition, the authors wish to thank David Hayes, Jason But, and Grenville Armitage, the authors of [DOI_10.1145_1496091.1496095], for their suggestions.

The authors also wish to thank Mohamed Boucadair for contributing the text related to the YANG module.

Authors' Addresses

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

Email: randall@lakerest.net

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

Email: tuexen@fh-muenster.de

Irene Rüngeler
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: i.ruengeler@fh-muenster.de

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

J. Touch
USC/ISI

April 24, 2015

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

Status of this Memo

Copyright Notice

Abstract

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

Table of Contents

1. Introduction

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

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

2. Conventions used in this document

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

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

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

3. History

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

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

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

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

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

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

4. Current Port Number Use

RFC6335 indicates three ranges of port number assignments:

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

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

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

5. What is a Port Number?

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

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

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

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

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

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

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

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

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

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

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

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

6. Conservation

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

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

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

6.1. Guiding Principles

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

- o A single assigned port number can support different functions over separate endpoint associations, determined using in-band information. An FTP data connection can transfer binary or text files, the latter translating line-terminators, as indicated in-band over the control port number [RFC959].

- o A single assigned port number can indicate the Dynamic port number(s) on which different capabilities are supported, as with passive-mode FTP [RFC959].

- o Several existing services can indicate the Dynamic port number(s) on which other services are supported, such as with mDNS and portmapper [RFC1833] [RFC6762] [RFC6763].

- o Copies of some existing services can be differentiated using in-band information (e.g., URIs in HTTP Host field and TLS Server Name Indication extension) [RFC7230] [RFC6066].

- o Services requiring varying performance properties can already be supported using separate endpoint associations (connections or other associations), each configured to support the desired properties. E.g., a high-speed and low-speed variant can be determined within the service using the same assigned port.

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

6.2. Firewall and NAT Considerations

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

Using Dynamic port numbers, or explicitly-indicated port numbers indicated in-band over another service (such as with FTP) often complicates firewall and NAT interactions [RFC959]. FTP over firewalls often requires direct support for deep-packet inspection

(to snoop for the Dynamic port number for the NAT to correctly map) or passive-mode FTP (in which both connections are opened from the client side).

7. Considerations for Requesting Port Number Assignments

Port numbers are assigned by IANA by a set of documented procedures [RFC6335]. The following section describes the steps users can take to help assist with responsible use of assigned port numbers, and with preparing an application for a port number assignment.

7.1. Is a port number assignment necessary?

First, it is useful to consider whether a port number assignment is required. In many cases, a new number assignment may not be needed, for example:

- Is this really a new service, or can an existing service suffice?

- Is this an experimental service [RFC3692]? If so, consider using the current experimental ports [RFC2780].

- Is this service independently useful? Some systems are composed from collections of different service capabilities, but not all component functions are useful as independent services. Port numbers are typically shared among the smallest independently-useful set of functions. Different service uses or properties can be supported in separate pairwise endpoint associations after an initial negotiation, e.g., to support software decomposition.

- Can this service use a Dynamic port number that is coordinated out-of-band, e.g.:

  - By explicit configuration of both endpoints.

  - By internal mechanisms within the same host (e.g., a configuration file, indicated within a URI, or using interprocess communication).

  - Using information exchanged on a related service: FTP, SIP, etc. [RFC959] [RFC3261].

  - Using an existing port discovery service: portmapper, mDNS, etc. [RFC1833] [RFC6762] [RFC6763].

There are a few good examples of reasons that more directly suggest that not only is a port number assignment not necessary, but it is directly counter-indicated:

- Assigned port numbers are not intended to differentiate performance variations within the same service, e.g., high-speed vs. ordinary speed. Performance variations can be supported within a single assigned port number in context of separate pairwise endpoint associations.

- Additional assigned port numbers are not intended to replicate an existing service. For example, if a device is configured to use a typical web browser then it the port number used for that service is a copy of the http service that is already assigned to port number 80 and does not warrant a new assignment. However, an automated system that happens to use HTTP framing - but is not primarily accessed by a browser - might be a new service. A good way to tell is "can an unmodified client of the existing service interact with the proposed service"? If so, that service would be a copy of an existing service and would not merit a new assignment.

- Assigned port numbers not intended for intra-machine communication. Such communication can already be supported by internal mechanisms (interprocess communication, shared memory, shared files, etc.). When Internet communication within a host is desired, the server can bind to a Dynamic port that is indicated to the client using these internal mechanisms.

- Separate assigned port numbers are not intended for insecure versions of existing (or new) secure services. A service that already requires security would be made more vulnerable by having the same capability accessible without security.

  Note that the converse is different, i.e., it can be useful to create a new, secure service that replicates an existing insecure service on a new port number assignment. This can be necessary when the existing service is not backward-compatible with security enhancements, such as the use of TLS [RFC5246] or DTLS [RFC6347].

- o Assigned port numbers are not intended for indicating different service versions. Version differentiation should be handled in-band, e.g., using a version number at the beginning of an association (e.g., connection or other transaction). This may not be possible with legacy assignments, but all new services should incorporate support for version indication.

Some services may not need assigned port numbers at all, e.g., SIP allows voice calls to use Dynamic ports [RFC3261]. Some systems can register services in the DNS, using SRV entries. These services can be discovered by a variety of means, including mDNS, or via direct query [RFC6762] [RFC6763]. In such cases, users can more easily request a SRV name, which are assigned first-come, first-served from a much larger namespace.

IANA assigns port numbers, but this assignment is typically used only for servers, i.e., the host that listens for incoming connections or other associations. Clients, i.e., hosts that initiate connections or other associations, typically refer to those assigned port numbers but do not need port number assignments for their endpoint.

Finally, an assigned port number is not a guarantee of exclusive use. Traffic for any service might appear on any port number, due to misconfiguration or deliberate misuse. Application and service designers are encouraged to validate traffic based on its content.

7.2. How Many Assigned Port Numbers?

As noted earlier, systems might require a single port number assignment, but rarely require multiple port numbers. There are a variety of known ways to reduce assigned port number consumption. Although some may be cumbersome or inefficient, they are nearly always preferable to consuming additional port number assignments.

Such techniques include:

- o Use of a discovery service, either a shared service (mDNS), or a discovery service for a given system [RFC6762] [RFC6763].

- o Multiplex packet types using in-band information, either on a per-message or per-connection basis. Such demultiplexing can even hand-off different messages and connections among different processes, such as is done with FTP [RFC959].

There are some cases where NAT and firewall traversal are significantly improved by having an assigned port number. Although

NAT traversal protocols supporting automatic configuration have been proposed and developed (e.g., STUN [RFC5389], TURN [RFC5766], and ICE [RFC5245]), not all application and service designers can rely on their presence as of yet.

In the past, some services were assigned multiple port numbers or sometimes fairly large port ranges (e.g., X11). This occurred for a variety of reasons: port number conservation was not as widely appreciated, assignments were not as ardently reviewed, etc. This no longer reflects current practice and such assignments are not considered to constitute a precedent for future assignments.

7.3. Picking an Assigned Port Number

Given a demonstrated need for a port number assignment, the next question is how to pick the desired port number. An application for a port number assignment does not need to include a desired port number; in that case, IANA will select from those currently available.

Users should consider whether the requested port number is important. For example, would an assignment be acceptable if IANA picked the port number value? Would a TCP (or other transport protocol) port number assignment be useful by itself? If so, a port number can be assigned to a service for one transport protocol where it is already (or can be subsequently) assigned to a different service for other transport protocols.

The most critical issue in picking a number is selecting the desired range, i.e., System vs. User port numbers. The distinction was intended to indicate a difference in privilege; originally, System port numbers required privileged (’root’) access, while User port numbers did not. That distinction has since blurred because some current systems do not limit access control to System port numbers and because some System services have been replicated on User numbers (e.g., IRC). Even so, System port number assignments have continued at an average rate of 3-4 per year over the past 7 years (2007-2013), indicating that the desire to keep this distinction continues.

As a result, the difference between System and User port numbers needs to be treated with caution. Developers are advised to treat services as if they are always run without privilege.

Even when developers seek a System port number assignment, it may be very difficult to obtain. System port number assignment requires IETF Review or IESG Approval and justification that both User and

Dynamic port number ranges are insufficient [RFC6335]. Thus this document recommends both:

>> Developers SHOULD NOT apply for System port number assignments because the increased privilege they are intended to provide is not always enforced.

>> System implementers SHOULD enforce the need for privilege for processes to listen on System port numbers.

At some future date, it might be useful to deprecate the distinction between System and User port numbers altogether. Services typically require elevated (’root’) privileges to bind to a System port number, but many such services go to great lengths to immediately drop those privileges just after connection or other association establishment to reduce the impact of an attack using their capabilities. Such services might be more securely operated on User port numbers than on System port numbers. Further, if System port numbers were no longer assigned, as of 2014 it would cost only 180 of the 1024 System values (17%), or 180 of the overall 49152 assigned (System and User) values (<0.04%).

7.4. Support for Security

Just as a service is a way to obtain information or processing from a host over a network, a service can also be the opening through which to compromise that host. Protecting a service involves security, which includes integrity protection, source authentication, privacy, or any combination of these capabilities. Security can be provided in a number of ways, and thus:

>> New services SHOULD support security capabilities, either directly or via a content protection such as TLS [RFC5246] or DTLS [RFC6347] or transport protection such as TCP-AO [RFC5925]. Insecure versions of new or existing secure services SHOULD be avoided because of the new vulnerability they create.

Secure versions of legacy services that are not already security-capable via in-band negotiations can be very useful. However, there is no IETF consensus on when separate ports should be used for secure and insecure variants of the same service [RFC2595] [RFC2817] [RFC6335]. The overall preference is for use of a single port, as noted in Section 6 of this document and Section 7.2 of [RFC6335], but the appropriate approach depends on the specific characteristics of the service. As a result:

>> When requesting both secure and insecure port assignments for the same service, justification is expected for the utility and safety of each port as an independent service (Section 6). Precedent (e.g., citing other protocols that use a separate insecure port) is inadequate justification by itself.

It’s also important to recognize that port number assignment is not itself a guarantee that traffic using that number provides the corresponding service, or that a given service is always offered only on its assigned port number. Port numbers are ultimately meaningful only between endpoints and any service can be run on any port. Thus:

>> Security SHOULD NOT rely on assigned port number distinctions alone; every service, whether secure or not, is likely to be attacked.

Applications for a new service that requires both a secure and insecure port may be found, on expert review, to be unacceptable, and may not be approved for allocation. Similarly, an application for a new port to support an insecure variant of an existing secure protocol may be found unacceptable. In both cases, the resulting security of the service in practice will be a significant consideration in the decision as to whether to assign an insecure port.

7.5. Support for Future Versions

Requests for assigned port numbers are expected to support multiple versions on the same assigned port number [RFC6335]. Versions are typically indicated in-band, either at the beginning of a connection or other association, or in each protocol message.

>> Version support SHOULD be included in new services rather than relying on different port number assignments for different versions.

>> Version numbers SHOULD NOT be included in either the service name or service description, to avoid the need to make additional port number assignments for future variants of a service.

Again, the assigned port number space is far too limited to be used as an indicator of protocol version or message type. Although this has happened in the past (e.g., for NFS), it should be avoided in new requests.

7.6. Transport Protocols

IANA assigns port numbers specific to one or more transport protocols, typically UDP [RFC768] and TCP [RFC793], but also SCTP [RFC4960], DCCP [RFC4340], and any other standard transport protocol. Originally, IANA port number assignments were concurrent for both UDP and TCP, and other transports were not indicated. However, to conserve the assigned port number space and to reflect increasing use of other transports, assignments are now specific only to the transport being used.

In general, a service should request assignments for multiple transports using the same service name and description on the same port number only when they all reflect essentially the same service. Good examples of such use are DNS and NFS, where the difference between the UDP and TCP services are specific to supporting each transport. E.g., the UDP variant of a service might add sequence numbers and the TCP variant of the same service might add in-band message delimiters. This document does not describe the appropriate selection of a transport protocol for a service.

>> Service names and descriptions for multiple transport port number assignments SHOULD match only when they describe the same service, excepting only enhancements for each supported transport.

When the services differ, it may be acceptable or preferable to use the same port number, but the service names and descriptions should be different for each transport/service pair, reflecting the differences in the services. E.g., if TCP is used for the basic control protocol and UDP for an alarm protocol, then the services might be "name-ctl" and "name-alarm". A common example is when TCP is used for a service and UDP is used to determine whether that service is active (e.g., via a unicast, broadcast, or multicast test message) [RFC1122]. IANA has, for several years, used the suffix "-disc" in service names to distinguish discovery services, such as are used to identify endpoints capable of a given service:

>> Names of discovery services SHOULD use an identifiable suffix; the suggestion is "-disc".

Some services are used for discovery, either in conjunction with a TCP service or as a stand-alone capability. Such services will be more reliable when using multicast rather than broadcast (over IPv4) because IP routers do not forward "all nodes" broadcasts (all 1’s, i.e., 255.255.255.255 for IPv4) and have not been required to support subnet-directed broadcasts since 1999 [RFC1812] [RFC2644].

This issue is relevant only for IPv4 because IPv6 does not support broadcast.

>> UDP over IPv4 multi-host services SHOULD use multicast rather than broadcast.

Designers should be very careful in creating services over transports that do not support congestion control or error recovery, notably UDP. There are several issues that should be considered in such cases, as summarized in Table 1 in [RFC5405]. In addition, the following recommendations apply to service design:

>> Services that use multipoint communication SHOULD be scalable, and SHOULD NOT rely solely on the efficiency of multicast transmission for scalability.

>> Services SHOULD NOT use UDP as a performance enhancement over TCP, e.g., to circumnavigate TCP’s congestion control.

7.7. When to Request an Assignment

Assignments are typically requested when a user has enough information to reasonably answer the questions in the IANA application. IANA applications typically take up to a few weeks to process, with some complex cases taking up to a month. The process typically involves a few exchanges between the IANA Ports Expert Review team and the applicant.

An application needs to include a description of the service, as well as to address key questions designed to help IANA determine whether the assignment is justified. The application should be complete and not refer solely to the Internet Draft, RFC, a website, or any other external documentation.

Services that are independently developed can be requested at any time, but are typically best requested in the last stages of design and initial experimentation, before any deployment has occurred that cannot easily be updated.

>> Users MUST NOT deploy implementations that use assigned port numbers prior their assignment by IANA.

>> Users MUST NOT deploy implementations that default to using the experimental System port numbers (1021 and 1022 [RFC4727]) outside a controlled environment where they can be updated with a subsequent assigned port [RFC3692].

Deployments that use unassigned port numbers before assignment complicate IANA management of the port number space. Keep in mind that this recommendation protects existing assignees, users of current services, and applicants for new assignments; it helps ensure that a desired number and service name are available when assigned. The list of currently unassigned numbers is just that - *currently* unassigned. It does not reflect pending applications. Waiting for an official IANA assignment reduces the chance that an assignment request will conflict with another deployed service.

Applications made through Internet Draft / RFC publication (in any stream) typically use a placeholder ("PORTNUM") in the text, and implementations use an experimental port number until a final assignment has been made [RFC6335]. That assignment is initially indicated in the IANA Considerations section of the document, which is tracked by the RFC Editor. When a document has been approved for publication, that request is forwarded to IANA for handling. IANA will make the new assignment accordingly. At that time, IANA may also request that the applicant fill out the application form on their website, e.g., when the RFC does not directly address the information expected as per [RFC6335]. "Early" assignments can be made when justified, e.g., for early interoperability testing, according to existing process [RFC7120] [RFC6335].

>> Users writing specifications SHOULD use symbolic names for port numbers and service names until an IANA assignment has been completed. Implementations SHOULD use experimental port numbers during this time, but those numbers MUST NOT be cited in documentation except as interim.

7.8. Squatting

"Squatting" describes the use of a number from the assignable range in deployed software without IANA assignment for that use, regardless of whether the number has been assigned or remains available for assignment. It is hazardous because IANA cannot track such usage and thus cannot avoid making legitimate assignments that conflict with such unauthorized usage.

Such "squatted" port numbers remain unassigned, and IANA retains the right to assign them when requested by other applicants. Application and service designers are reminded that is never appropriate to use port numbers that have not been directly assigned [RFC6335]. In particular, any unassigned code from the assigned ranges will be assigned by IANA, and any conflict will be easily resolved as the protocol designer’s fault once that happens (because they would not be the assignee). This may reflect in the public’s judgment on the

quality of their expertise and cooperation with the Internet community.

Regardless, there are numerous services that have squatted on such numbers that are in widespread use. Designers who are using such port numbers are encouraged to apply for an assignment. Note that even widespread de-facto use may not justify a later IANA assignment of that value, especially if either the value has already been assigned to a legitimate applicant or if the service would not qualify for an assignment of its own accord.

7.9. Other Considerations

As noted earlier, System port numbers should be used sparingly, and it is better to avoid them altogether. This avoids the potentially incorrect assumption that the service on such port numbers run in a privileged mode.

Assigned port numbers are not intended to be changed; this includes the corresponding service name. Once deployed, it can be very difficult to recall every implementation, so the assignment should be retained. However, in cases where the current assignee of a name or number has reasonable knowledge of the impact on such uses, and is willing to accept that impact, the name or number of an assignment can be changed [RFC6335]

Aliases, or multiple service names for the same assigned port number, are no longer considered appropriate [RFC6335].

8. Security Considerations

This document focuses on the issues arising when designing services that require new port assignments. Section 7.4 addresses the security and security-related issues of that interaction.

When designing a secure service, the use of TLS [RFC5246], DTLS [RFC6347], or TCP-AO [RFC5925] mechanisms that protect transport protocols or their contents is encouraged. It may not be possible to use IPsec [RFC4301] in similar ways because of the different relationship between IPsec and port numbers and because applications may not be aware of IPsec protections.

This document reminds application and service designers that port numbers do not protect against denial of service attack or guarantee that traffic should be trusted. Using assigned numbers for port filtering isn’t a substitute for authentication, encryption, and integrity protection. The port number alone should not be used to

avoid denial of service attacks or to manage firewall traffic because the use of port numbers is not regulated or validated.

The use of assigned port numbers is the antithesis of privacy because they are intended to explicitly indicate the desired application or service. Strictly, port numbers are meaningful only at the endpoints, so any interpretation elsewhere in the network can be arbitrarily incorrect. However, those numbers can also expose information about available services on a given host. This information can be used by intermediate devices to monitor and intercept traffic as well as to potentially identify key endpoint software properties ("fingerprinting"), which can be used to direct other attacks.

9. IANA Considerations

The entirety of this document focuses on suggestions that help ensure the conservation of port numbers and provide useful hints for issuing informative requests thereof.

10. References

10.1. Normative References

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

[RFC2780] Bradner, S., and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.

[RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3962, Jan. 2004.

[RFC4727] Fenner, B., "Experimental Values in IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, November 2006.

[RFC5246] Dierks, T., and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5405] Eggert, L., and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, Nov. 2008.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

[RFC6335] Cotton, M., L. Eggert, J. Touch, M. Westerlund, and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

[RFC6347] Rescorla, E., and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

10.2. Informative References

[IEN112] Postel, J., "Transmission Control Protocol", IEN 112, August 1979.

[RFC33] Crocker, S., "New Host-Host Protocol", RFC 33 February 1970.

[RFC37] Crocker, S., "Network Meeting Epilogue", RFC 37, March 1970.

[RFC38] Wolfe, S., "Comments on Network Protocol from NWG/RFC #36", RFC 38, March 1970.

[RFC48] Postel, J., and S. Crocker, "Possible protocol plateau", RFC 48, April 1970.

[RFC61] Walden, D., "Note on Interprocess Communication in a Resource Sharing Computer Network", RFC 61, July 1970.

[RFC76] Bouknight, J., J. Madden, and G. Grossman, "Connection by name: User oriented protocol", RFC 76, October 1970.

[RFC333] Bressler, R., D. Murphy, and D. Walden. "Proposed experiment with a Message Switching Protocol", RFC 333, May 1972.

[RFC739] Postel, J., "Assigned numbers", RFC 739, November 1977.

[RFC758] Postel, J., "Assigned numbers", RFC 758, August 1979.

[RFC768] Postel, J., "User Datagram Protocol", RFC 768, August 1980.

[RFC793] Postel, J., "Transmission Control Protocol" RFC 793, September 1981

[RFC820] Postel, J., "Assigned numbers", RFC 820, August 1982.

[RFC900] Reynolds, J., and J. Postel, "Assigned numbers", RFC 900, June 1984.

[RFC959] Postel, J., and J. Reynolds, "FILE TRANSFER PROTOCOL (FTP)", RFC 959, October 1985.

[RFC1122] Braden, B. (Ed.), "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989.

[RFC1340] Reynolds, J., and J. Postel, "Assigned numbers", RFC 1340, July 1992.

[RFC1700] Reynolds, J., and J. Postel, "Assigned numbers", RFC 1700, October 1994.

[RFC1812] Baker, F. (Ed.), "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2", RFC 1833, August 1995.

[RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999.

[RFC2644] Senie, D., "Changing the Default for Directed Broadcasts in Routers", RFC 2644, August 1999.

[RFC2817] Khare, R., and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.

[RFC3232] Reynolds, J. (Ed.), "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, January 2002.

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

[RFC4301] Kent, S., and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4340] Kohler, E., M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4960] Stewart, R. (Ed.), "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5389] Rosenberg, J., R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT", RFC 5389, October 2008.

[RFC5766] Mahy, R., P. Matthews, and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6762] Cheshire, S., and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[RFC6763] Cheshire, S., and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, January 2014.

[RFC7230] Fielding, R., (Ed.), and J. Reshke, (Ed.), "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

11. Acknowledgments

This work benefitted from the feedback from David Black, Lars Eggert, Gorry Fairhurst, and Eliot Lear, as well as discussions of the IETF TSVWG WG.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292-6695
U.S.A.

Phone: +1 (310) 448-9151
EMail: touch@isi.edu

Transport Area Working Group
Internet-Draft
Obsoletes: 5405 (if approved)
Intended status: Best Current Practice
Expires: April 17, 2017

L. Eggert
NetApp
G. Fairhurst
University of Aberdeen
G. Shepherd
Cisco Systems

October 14, 2016

UDP Usage Guidelines
draft-ietf-tsvwg-rfc5405bis-19

Abstract

The User Datagram Protocol (UDP) provides a minimal message-passing transport that has no inherent congestion control mechanisms. This document provides guidelines on the use of UDP for the designers of applications, tunnels and other protocols that use UDP. Congestion control guidelines are a primary focus, but the document also provides guidance on other topics, including message sizes, reliability, checksums, middlebox traversal, the use of ECN, DSCPs, and ports.

Because congestion control is critical to the stable operation of the Internet, applications and other protocols that choose to use UDP as an Internet transport must employ mechanisms to prevent congestion collapse and to establish some degree of fairness with concurrent traffic. They may also need to implement additional mechanisms, depending on how they use UDP.

Some guidance is also applicable to the design of other protocols (e.g., protocols layered directly on IP or via IP-based tunnels), especially when these protocols do not themselves provide congestion control.

This document obsoletes RFC5405 and adds guidelines for multicast UDP usage.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The User Datagram Protocol (UDP) [RFC0768] provides a minimal, unreliable, best-effort, message-passing transport to applications and other protocols (such as tunnels) that desire to operate over IP. Both are simply called "applications" in the remainder of this document.

Compared to other transport protocols, UDP and its UDP-Lite variant [RFC3828] are unique in that they do not establish end-to-end connections between communicating end systems. UDP communication consequently does not incur connection establishment and teardown overheads, and there is minimal associated end system state. Because of these characteristics, UDP can offer a very efficient communication transport to some applications.

A second unique characteristic of UDP is that it provides no inherent congestion control mechanisms. On many platforms, applications can send UDP datagrams at the line rate of the platform's link interface, which is often much greater than the available end-to-end path capacity, and doing so contributes to congestion along the path. [RFC2914] describes the best current practice for congestion control in the Internet. It identifies two major reasons why congestion control mechanisms are critical for the stable operation of the Internet:

1. The prevention of congestion collapse, i.e., a state where an increase in network load results in a decrease in useful work done by the network.

2. The establishment of a degree of fairness, i.e., allowing multiple flows to share the capacity of a path reasonably equitably.

Because UDP itself provides no congestion control mechanisms, it is up to the applications that use UDP for Internet communication to employ suitable mechanisms to prevent congestion collapse and establish a degree of fairness. [RFC2309] discusses the dangers of congestion-unresponsive flows and states that "all UDP-based streaming applications should incorporate effective congestion avoidance mechanisms." [RFC7567] reaffirms this statement. This is an important requirement, even for applications that do not use UDP for streaming. In addition, congestion-controlled transmission is of benefit to an application itself, because it can reduce self-induced packet loss, minimize retransmissions, and hence reduce delays. Congestion control is essential even at relatively slow transmission

rates. For example, an application that generates five 1500-byte UDP datagrams in one second can already exceed the capacity of a 56 Kb/s path. For applications that can operate at higher, potentially unbounded data rates, congestion control becomes vital to prevent congestion collapse and establish some degree of fairness. Section 3 describes a number of simple guidelines for the designers of such applications.

A UDP datagram is carried in a single IP packet and is hence limited to a maximum payload of 65,507 bytes for IPv4 and 65,527 bytes for IPv6. The transmission of large IP packets usually requires IP fragmentation. Fragmentation decreases communication reliability and efficiency and should be avoided. IPv6 allows the option of transmitting large packets ("jumbograms") without fragmentation when all link layers along the path support this [RFC2675]. Some of the guidelines in Section 3 describe how applications should determine appropriate message sizes. Other sections of this document provide guidance on reliability, checksums, middlebox traversal and use of multicast.

This document provides guidelines and recommendations. Although most UDP applications are expected to follow these guidelines, there do exist valid reasons why a specific application may decide not to follow a given guideline. In such cases, it is RECOMMENDED that application designers cite the respective section(s) of this document in the technical specification of their application or protocol and explain their rationale for their design choice.

[RFC5405] was scoped to provide guidelines for unicast applications only, whereas this document also provides guidelines for UDP flows that use IP anycast, multicast, broadcast, and applications that use UDP tunnels to support IP flows.

Finally, although this document specifically refers to usage of UDP, the spirit of some of its guidelines also applies to other message-passing applications and protocols (specifically on the topics of congestion control, message sizes, and reliability). Examples include signaling, tunnel, or control applications that choose to run directly over IP by registering their own IP protocol number with IANA. This document is expected to provide useful background reading to the designers of such applications and protocols.

2. Terminology

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

3. UDP Usage Guidelines

Internet paths can have widely varying characteristics, including transmission delays, available bandwidths, congestion levels, reordering probabilities, supported message sizes, or loss rates. Furthermore, the same Internet path can have very different conditions over time. Consequently, applications that may be used on the Internet MUST NOT make assumptions about specific path characteristics. They MUST instead use mechanisms that let them operate safely under very different path conditions. Typically, this requires conservatively probing the current conditions of the Internet path they communicate over to establish a transmission behavior that it can sustain and that is reasonably fair to other traffic sharing the path.

These mechanisms are difficult to implement correctly. For most applications, the use of one of the existing IETF transport protocols is the simplest method of acquiring the required mechanisms. Doing so also avoids issues that protocols using a new IP protocol number face when being deployed over the Internet, where middleboxes that only support TCP and UDP are sometimes present. Consequently, the RECOMMENDED alternative to the UDP usage described in the remainder of this section is the use of an IETF transport protocol such as TCP [RFC0793], Stream Control Transmission Protocol (SCTP) [RFC4960], and SCTP Partial Reliability Extension (SCTP-PR) [RFC3758], or Datagram Congestion Control Protocol (DCCP) [RFC4340] with its different congestion control types [RFC4341][RFC4342][RFC5622], or transport protocols specified by the IETF in the future. (UDP-encapsulated SCTP [RFC6951] and DCCP [RFC6773] can offer support for traversing firewalls and other middleboxes where the native protocols are not supported.)

If used correctly, these more fully-featured transport protocols are not as "heavyweight" as often claimed. For example, the TCP algorithms have been continuously improved over decades, and have reached a level of efficiency and correctness that custom application-layer mechanisms will struggle to easily duplicate. In addition, many TCP implementations allow connections to be tuned by an application to its purposes. For example, TCP’s "Nagle" algorithm [RFC1122] can be disabled, improving communication latency at the expense of more frequent -- but still congestion-controlled -- packet transmissions. Another example is the TCP SYN cookie mechanism [RFC4987], which is available on many platforms. TCP with SYN cookies does not require a server to maintain per-connection state until the connection is established. TCP also requires the end that closes a connection to maintain the TIME-WAIT state that prevents delayed segments from one connection instance from interfering with a later one. Applications that are aware of and designed for this

behavior can shift maintenance of the TIME-WAIT state to conserve resources by controlling which end closes a TCP connection [FABER]. Finally, TCP's built-in capacity-probing and awareness of the maximum transmission unit supported by the path (PMTU) results in efficient data transmission that quickly compensates for the initial connection setup delay, in the case of transfers that exchange more than a few segments.

3.1. Congestion Control Guidelines

If an application or protocol chooses not to use a congestion-controlled transport protocol, it SHOULD control the rate at which it sends UDP datagrams to a destination host, in order to fulfill the requirements of [RFC2914]. It is important to stress that an application SHOULD perform congestion control over all UDP traffic it sends to a destination, independently from how it generates this traffic. For example, an application that forks multiple worker processes or otherwise uses multiple sockets to generate UDP datagrams SHOULD perform congestion control over the aggregate traffic.

Several approaches to perform congestion control are discussed in the remainder of this section. The section describes generic topics with an intended emphasis on unicast and anycast [RFC1546] usage. Not all approaches discussed below are appropriate for all UDP-transmitting applications. Section 3.1.2 discusses congestion control options for applications that perform bulk transfers over UDP. Such applications can employ schemes that sample the path over several subsequent round-trips during which data is exchanged to determine a sending rate that the path at its current load can support. Other applications only exchange a few UDP datagrams with a destination. Section 3.1.3 discusses congestion control options for such "low data-volume" applications. Because they typically do not transmit enough data to iteratively sample the path to determine a safe sending rate, they need to employ different kinds of congestion control mechanisms. Section 3.1.11 discusses congestion control considerations when UDP is used as a tunneling protocol. Section 4 provides additional recommendations for broadcast and multicast usage.

It is important to note that congestion control should not be viewed as an add-on to a finished application. Many of the mechanisms discussed in the guidelines below require application support to operate correctly. Application designers need to consider congestion control throughout the design of their application, similar to how they consider security aspects throughout the design process.

In the past, the IETF has also investigated integrated congestion control mechanisms that act on the traffic aggregate between two hosts, i.e., a framework such as the Congestion Manager [RFC3124], where active sessions may share current congestion information in a way that is independent of the transport protocol. Such mechanisms have currently failed to see deployment, but would otherwise simplify the design of congestion control mechanisms for UDP sessions, so that they fulfill the requirements in [RFC2914].

3.1.1. Protocol Timer Guidelines

Understanding the latency between communicating endpoints is usually a crucial part of effective congestion control implementations for protocols and applications. Latency estimation can be used in a number of protocol functions, such as calculating a congestion-controlled transmission rate, triggering retransmission, and detecting packet loss. Additional protocol functions, for example, determining an interval for probing a path, determining an interval between keep-alive messages, determining an interval for measuring the quality of experience, or determining if a remote endpoint has responded to a request to perform an action typically operate over longer timescales than congestion control and therefore are not covered in this section.

The general recommendation in this document is that applications SHOULD leverage existing congestion control techniques and the latency estimators specified therein (see next subsection). The following guidelines are provided for applications that need to design their own latency estimation mechanisms.

The guidelines are framed in terms of "latency" and not "round-trip time" because some situations require characterizing only the network-based latency (e.g., TCP-Friendly Rate Control [RFC5348]), while other cases necessitate inclusion of the time required by the remote endpoint to provide feedback (e.g., developing an understanding of when to retransmit a message).

The latency between endpoints is generally a dynamic property. Therefore, estimates SHOULD represent some sort of averaging of multiple recent measurement samples to account for variance. Leveraging an Exponentially Weighted Moving Average (EWMA) has proven useful for this purpose (e.g., in TCP [RFC6298] and TCP-Friendly Rate Control (TFRC) [RFC5348]).

Independent latency estimates SHOULD be maintained for each destination with which an endpoint communicates.

Latency samples MUST NOT be derived from ambiguous transactions. The canonical example is in a protocol that retransmits data, but subsequently cannot determine which copy is being acknowledged. This ambiguity makes correct computation of the latency problematic. See the discussion of Karn's algorithm in [RFC6298]. This requirement ensures a sender establishes a sound estimate of the latency without relying on mis-leading measurements.

When a latency estimate is used to arm a timer that provides loss detection - with or without retransmission - expiry of the timer MUST be interpreted as an indication of congestion in the network, causing the sending rate to be adapted to a safe conservative rate (e.g., TCP collapses the congestion window to one segment [RFC5681]).

Some applications require an initial latency estimate before the latency between endpoints can be empirically sampled. For instance, when arming a retransmission timer an initial value is needed to protect the messages sent before the endpoints sample the latency. This initial latency estimate SHOULD generally be as conservative (large) as possible for the given application. For instance, in the absence of any knowledge about the latency of a path, TCP requires the initial Retransmission Timeout (RTO) to be set to no less than 1 second [RFC6298]. UDP applications SHOULD similarly use an initial latency estimate of 1 second. Values shorter than 1 second can be problematic (see the data analysis in the appendix of [RFC6298]).

3.1.2. Bulk Transfer Applications

Applications that perform bulk transmission of data to a peer over UDP, i.e., applications that exchange more than a few UDP datagrams per round-trip time (RTT), SHOULD implement TFRC [RFC5348], window-based TCP-like congestion control, or otherwise ensure that the application complies with the congestion control principles.

TFRC has been designed to provide both congestion control and fairness in a way that is compatible with the IETF's other transport protocols. If an application implements TFRC, it need not follow the remaining guidelines in Section 3.1.2, because TFRC already addresses them, but SHOULD still follow the remaining guidelines in the subsequent subsections of Section 3.

Bulk transfer applications that choose not to implement TFRC or TCP-like windowing SHOULD implement a congestion control scheme that results in bandwidth (capacity) use that competes fairly with TCP within an order of magnitude.

Section 2 of [RFC3551] suggests that applications SHOULD monitor the packet loss rate to ensure that it is within acceptable parameters.

Packet loss is considered acceptable if a TCP flow across the same network path under the same network conditions would achieve an average throughput, measured on a reasonable timescale, that is not less than that of the UDP flow. The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in timescale and throughput. The recommendations for managing timers specified in Section 3.1.1 also apply.

Finally, some bulk transfer applications may choose not to implement any congestion control mechanism and instead rely on transmitting across reserved path capacity (see Section 3.1.9). This might be an acceptable choice for a subset of restricted networking environments, but is by no means a safe practice for operation over the wider Internet. When the UDP traffic of such applications leaks out into unprovisioned Internet paths, it can significantly degrade the performance of other traffic sharing the path and even result in congestion collapse. Applications that support an uncontrolled or unadaptive transmission behavior SHOULD NOT do so by default and SHOULD instead require users to explicitly enable this mode of operation, and they SHOULD verify that sufficient path capacity has been reserved for them.

3.1.3. Low Data-Volume Applications

When applications that at any time exchange only a few UDP datagrams with a destination implement TFRC or one of the other congestion control schemes in Section 3.1.2, the network sees little benefit, because those mechanisms perform congestion control in a way that is only effective for longer transmissions.

Applications that at any time exchange only a few UDP datagrams with a destination SHOULD still control their transmission behavior by not sending on average more than one UDP datagram per RTT to a destination. Similar to the recommendation in [RFC1536], an application SHOULD maintain an estimate of the RTT for any destination with which it communicates using the methods specified in Section 3.1.1.

Some applications cannot maintain a reliable RTT estimate for a destination. These applications do not need to or are unable to use protocol timers to measure the RTT (Section 3.1.1). Two cases can be identified:

1. The first case is that of applications that exchange too few UDP datagrams with a peer to establish a statistically accurate RTT estimate, but can monitor the reliability of transmission (Section 3.3). Such applications MAY use a predetermined transmission interval that is exponentially backed-off when

packets are found to be lost. TCP specifies an initial value of 1 second [RFC6298], which is also RECOMMENDED as an initial value for UDP applications. Some low data-volume applications, e.g., SIP [RFC3261] and GIST [RFC5971] use an interval of 500 ms, and shorter values are likely problematic in many cases. As in the previous case, note that the initial timeout is not the maximum possible timeout, see Section 3.1.1.

2. A second case of applications cannot maintain an RTT estimate for a destination, because the destination does not send return traffic. Such applications SHOULD NOT send more than one UDP datagram every 3 seconds, and SHOULD use an even less aggressive rate when possible. Shorter values are likely problematic in many cases. Note that the sending rate in this case must be more conservative than in the previous cases, because the lack of return traffic prevents the detection of packet loss, i.e., congestion, and the application therefore cannot perform exponential back-off to reduce load.

3.1.4. Applications supporting bidirectional communications

Applications that communicate bidirectionally SHOULD employ congestion control for both directions of the communication. For example, for a client-server, request-response-style application, clients SHOULD congestion-control their request transmission to a server, and the server SHOULD congestion-control its responses to the clients. Congestion in the forward and reverse direction is uncorrelated, and an application SHOULD either independently detect and respond to congestion along both directions, or limit new and retransmitted requests based on acknowledged responses across the entire round-trip path.

3.1.5. Implications of RTT and Loss Measurements on Congestion Control

Transports such as TCP, SCTP and DCCP provide timely detection of congestion that results in an immediate reduction of their maximum sending rate when congestion is experienced. This reaction is typically completed 1-2 RTTs after loss/congestion is encountered. Applications using UDP SHOULD implement a congestion control scheme that provides a prompt reaction to signals indicating congestion (e.g., by reducing the rate within the next RTT following a congestion signal).

The operation of a UDP congestion control algorithm can be very different to the way TCP operates. This includes congestion controls that respond on timescales that fit applications that cannot usefully work within the "change rate every RTT" model of TCP. Applications that experience a low or varying RTT are particularly vulnerable to

sampling errors (e.g., due to measurement noise, or timer accuracy). This suggests the need to average loss/congestion and RTT measurements over a longer interval, however this also can contribute additional delay in detecting congestion. Some applications may not react by reducing their sending rate immediately for various reasons, including: RTT and loss measurements are only made periodically (e.g., using RTCP), additional time is required to filter information, or the application is only able to change its sending rate at predetermined interval (e.g., some video codecs).

When designing a congestion control algorithm, the designer therefore needs to consider the total time taken to reduce the load following a lack of feedback or a congestion event. An application where the most recent RTT measurement is smaller than the actual RTT or the measured loss rate is smaller than the current rate, can result in over estimating the available capacity. Such over estimation can result in a sending rate that creates congestion to the application or other flows sharing the path capacity, and can contribute to congestion collapse - both of these need to be avoided.

A congestion control designed for UDP SHOULD respond as quickly as possible when it experiences congestion, and SHOULD take into account both the loss rate and the response time when choosing a new rate. The implemented congestion control scheme SHOULD result in bandwidth (capacity) use that is comparable to that of TCP within an order of magnitude, so that it does not starve other flows sharing a common bottleneck.

3.1.6. Burst Mitigation and Pacing

UDP applications SHOULD provide mechanisms to regulate the bursts of transmission that the application may send to the network. Many TCP and SCTP implementations provide mechanisms that prevent a sender from generating long bursts at line-rate, since these are known to induce early loss to applications sharing a common network bottleneck. The use of pacing with TCP [ALLMAN] has also been shown to improve the coexistence of TCP flows with other flows. The need to avoid excessive transmission bursts is also noted in specifications for applications (e.g., [RFC7143]).

Even low data-volume UDP flows may benefit from packet pacing, e.g., an application that sends three copies of a packet to improve robustness to loss is RECOMMENDED to pace out those three packets over several RTTs, to reduce the probability that all three packets will be lost due to the same congestion event (or other event, such as burst corruption).

3.1.7. Explicit Congestion Notification

Internet applications can use Explicit Congestion Notification (ECN) [RFC3168] to gain benefits for the services they support [I-D.ietf-aqm-ecn-benefits].

Internet transports, such as TCP, provide a set of mechanisms that are needed to utilize ECN. ECN operates by setting an ECN-capable codepoint (ECT(0) or ECT(1)) in the IP header of packets that are sent. This indicates to ECN-capable network devices (routers, and other devices) that they may mark (set the congestion experienced, CE codepoint), rather than drop the IP packet as a signal of incipient congestion.

UDP applications can also benefit from enabling ECN, providing that the API supports ECN and that they implement the required protocol mechanisms to support ECN.

The set of mechanisms required for an application to use ECN over UDP are:

- o A sender MUST provide a method to determine (e.g., negotiate) that the corresponding application is able to provide ECN feedback using a compatible ECN method.

- o A receiver that enables the use of ECN for a UDP port MUST check the ECN field at the receiver for each UDP datagram that it receives on this port.

- o The receiving application needs to provide feedback of congestion information to the sending application. This MUST report the presence of datagrams received with a CE-mark by providing a mechanism to feed this congestion information back to the sending application. The feedback MAY also report the presence of ECT(1) and ECT(0)/Not-ECT packets [RFC7560]. ([RFC3168] and [RFC7560] specify methods for TCP.)

- o An application sending ECN-capable datagrams MUST provide an appropriate congestion reaction when it receives feedback indicating that congestion has been experienced. This ought to result in reduction of the sending rate by the UDP congestion control method (see Section 3.1) that is not less than the reaction of TCP under equivalent conditions.

- o A sender SHOULD detect network paths that do not support the ECN field correctly. When detected they need to either conservatively react to congestion or even fall back to not using ECN [I-D.ietf-aqm-ecn-benefits]. This method needs to be robust to

changes within the network path that may occur over the lifetime of a session.

- o A sender is encouraged to provide a mechanism to detect and react appropriately to misbehaving receivers that fail to report CE-marked packets [I-D.ietf-aqm-ecn-benefits].

[RFC6679] provides guidance and an example of this support, by describing a method to allow ECN to be used for UDP-based applications using the Real-Time Protocol (RTP). Applications that cannot provide this set of mechanisms, but wish to gain the benefits of using ECN, are encouraged to use a transport protocol that already supports ECN (such as TCP).

3.1.8. Differentiated Services Model

An application using UDP can use the differentiated services (DiffServ) Quality of Service (QoS) framework. To enable differentiated services processing, a UDP sender sets the Differentiated Services Code Point (DSCP) field [RFC2475] in packets sent to the network. Normally, a UDP source/destination port pair will set a single DSCP value for all packets belonging to a flow, but multiple DSCPs can be used as described later in this section. A DSCP may be chosen from a small set of fixed values (the class selector code points), or from a set of recommended values defined in the Per Hop Behavior (PHB) specifications, or from values that have purely local meanings to a specific network that supports DiffServ. In general, packets may be forwarded across multiple networks between source and destination.

In setting a non-default DSCP value, an application must be aware that DSCP markings may be changed or removed between the traffic source and destination. This has implications on the design of applications that use DSCPs. Specifically, applications SHOULD be designed to not rely on implementation of a specific network treatment, they need instead to implement congestion control methods to determine if their current sending rate is inducing congestion in the network.

[RFC7657] describes the implications of using DSCPs and provides recommendations on using multiple DSCPs within a single network five-tuple (source and destination addresses, source and destination ports, and the transport protocol used, in this case, UDP or UDP-Lite), and particularly the expected impact on transport protocol interactions, with congestion control or reliability functionality (e.g., retransmission, reordering). Use of multiple DSCPs can result in reordering by increasing the set of network forwarding resources

used by a sender. It can also increase exposure to resource depletion or failure.

3.1.9. QoS, Pre-Provisioned or Reserved Capacity

The IETF usually specifies protocols for use within the Best Effort General Internet. Sometimes it is relevant to specify protocols with a different applicability. An application using UDP can use the integrated services QoS framework. This framework is usually made available within controlled environments (e.g., within a single administrative domain or bilaterally agreed connection between domains). Applications intended for the Internet SHOULD NOT assume that QoS mechanisms are supported by the networks they use, and therefore need to provide congestion control, error recovery, etc. in case the actual network path does not provide provisioned service.

Some UDP applications are only expected to be deployed over network paths that use pre-provisioned capacity or capacity reserved using dynamic provisioning, e.g., through the Resource Reservation Protocol (RSVP). Multicast applications are also used with pre-provisioned capacity (e.g., IPTV deployments within access networks). These applications MAY choose not to implement any congestion control mechanism and instead rely on transmitting only on paths where the capacity is provisioned and reserved for this use. This might be an acceptable choice for a subset of restricted networking environments, but is by no means a safe practice for operation over the wider Internet. Applications that choose this option SHOULD carefully and in detail describe the provisioning and management procedures that result in the desired containment.

Applications that support an uncontrolled or unadaptive transmission behavior SHOULD NOT do so by default and SHOULD instead require users to explicitly enable this mode of operation.

Applications designed for use within a controlled environment (see Section 3.6) may be able to exploit network management functions to detect whether they are causing congestion, and react accordingly. If the traffic of such applications leaks out into unprovisioned Internet paths, it can significantly degrade the performance of other traffic sharing the path and even result in congestion collapse. Protocols designed for such networks SHOULD provide mechanisms at the network edge to prevent leakage of traffic into unprovisioned Internet paths (e.g., [RFC7510]). To protect other applications sharing the same path, applications SHOULD also deploy an appropriate circuit breaker, as described in Section 3.1.10.

An IETF specification targeting a controlled environment is expected to provide an applicability statement that restricts the application to the controlled environment (see Section 3.6).

3.1.10. Circuit Breaker Mechanisms

A transport circuit breaker is an automatic mechanism that is used to estimate the congestion caused by a flow, and to terminate (or significantly reduce the rate of) the flow when excessive congestion is detected [I-D.ietf-tsvwg-circuit-breaker]. This is a safety measure to prevent congestion collapse (starvation of resources available to other flows), essential for an Internet that is heterogeneous and for traffic that is hard to predict in advance.

A circuit breaker is intended as a protection mechanism of last resort. Under normal circumstances, a circuit breaker should not be triggered; it is designed to protect things when there is severe overload. The goal is usually to limit the maximum transmission rate that reflects the available capacity of a network path. Circuit breakers can operate on individual UDP flows or traffic aggregates, e.g., traffic sent using a network tunnel.

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

Applications used in the general Internet SHOULD implement a transport circuit breaker if they do not implement congestion control or operate a low volume data service (see Section 3.6). All applications MAY implement a transport circuit breaker [I-D.ietf-tsvwg-circuit-breaker] and are encouraged to consider implementing at least a slow-acting transport circuit breaker to provide a protection of last resort for their network traffic.

3.1.11. UDP Tunnels

One increasingly popular use of UDP is as a tunneling protocol [I-D.ietf-intarea-tunnels], where a tunnel endpoint encapsulates the packets of another protocol inside UDP datagrams and transmits them to another tunnel endpoint, which decapsulates the UDP datagrams and forwards the original packets contained in the payload. One example of such a protocol is Teredo [RFC4380]. Tunnels establish virtual links that appear to directly connect locations that are distant in the physical Internet topology and can be used to create virtual (private) networks. Using UDP as a tunneling protocol is attractive when the payload protocol is not supported by middleboxes that may exist along the path, because many middleboxes support transmission using UDP.

Well-implemented tunnels are generally invisible to the endpoints that happen to transmit over a path that includes tunneled links. On the other hand, to the routers along the path of a UDP tunnel, i.e., the routers between the two tunnel endpoints, the traffic that a UDP tunnel generates is a regular UDP flow, and the encapsulator and decapsulator appear as regular UDP-sending and -receiving applications. Because other flows can share the path with one or more UDP tunnels, congestion control needs to be considered.

Two factors determine whether a UDP tunnel needs to employ specific congestion control mechanisms -- first, whether the payload traffic is IP-based; second, whether the tunneling scheme generates UDP traffic at a volume that corresponds to the volume of payload traffic carried within the tunnel.

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

However, if the IP traffic in the tunnel is known to not be congestion-controlled, additional measures are RECOMMENDED to limit the impact of the tunneled traffic on other traffic sharing the path. For the specific case of a tunnel that carries IP multicast traffic, see Section 4.1.

The following guidelines define these possible cases in more detail:

1. A tunnel generates UDP traffic at a volume that corresponds to the volume of payload traffic, and the payload traffic is IP-based and congestion-controlled.

   This is arguably the most common case for Internet tunnels. In this case, the UDP tunnel SHOULD NOT employ its own congestion control mechanism, because congestion losses of tunneled traffic will already trigger an appropriate congestion response at the original senders of the tunneled traffic. A circuit breaker mechanism may provide benefit by controlling the envelope of the aggregated traffic.

   Note that this guideline is built on the assumption that most IP-based communication is congestion-controlled. If a UDP tunnel is used for IP-based traffic that is known to not be congestion-controlled, the next set of guidelines applies.

2. A tunnel generates UDP traffic at a volume that corresponds to the volume of payload traffic, and the payload traffic is not known to be IP-based, or is known to be IP-based but not congestion-controlled.

   This can be the case, for example, when some link-layer protocols are encapsulated within UDP (but not all link-layer protocols; some are congestion-controlled). Because it is not known that congestion losses of tunneled non-IP traffic will trigger an appropriate congestion response at the senders, the UDP tunnel SHOULD employ an appropriate congestion control mechanism or circuit breaker mechanism designed for the traffic it carries. Because tunnels are usually bulk-transfer applications as far as the intermediate routers are concerned, the guidelines in Section 3.1.2 apply.

3. A tunnel generates UDP traffic at a volume that does not correspond to the volume of payload traffic, independent of whether the payload traffic is IP-based or congestion-controlled.

   Examples of this class include UDP tunnels that send at a constant rate, increase their transmission rates under loss, for example, due to increasing redundancy when Forward Error Correction is used, or are otherwise unconstrained in their transmission behavior. These specialized uses of UDP for tunneling go beyond the scope of the general guidelines given in this document. The implementer of such specialized tunnels SHOULD carefully consider congestion control in the design of their tunneling mechanism and SHOULD consider use of a circuit breaker mechanism.

The type of encapsulated payload might be identified by a UDP port; identified by an Ethernet Type or IP protocol number. A tunnel SHOULD provide mechanisms to restrict the types of flows that may be carried by the tunnel. For instance, a UDP tunnel designed to carry IP needs to filter out non-IP traffic at the ingress. This is particularly important when a generic tunnel encapsulation is used (e.g., one that encapsulates using an EtherType value). Such tunnels SHOULD provide a mechanism to restrict the types of traffic that are allowed to be encapsulated for a given deployment (see [I-D.ietf-intarea-tunnels]).

Designing a tunneling mechanism requires significantly more expertise than needed for many other UDP applications, because tunnels are usually intended to be transparent to the endpoints transmitting over them, so they need to correctly emulate the behavior of an IP link [I-D.ietf-intarea-tunnels], e.g.:

- o Requirements for tunnels that carry or encapsulate using ECN code points [RFC6040].

- o Usage of the IP DSCP field by tunnel endpoints [RFC2983].

- o Encapsulation considerations in the design of tunnels [I-D.ietf-rtgwg-dt-encap].

- o Usage of ICMP messages [I-D.ietf-intarea-tunnels].

- o Handling of fragmentation and packet size for tunnels [I-D.ietf-intarea-tunnels].

- o Source port usage for tunnels designed to support equal cost multipath (ECMP) routing (see Section 5.1.1).

- o Guidance on the need to protect headers [I-D.ietf-intarea-tunnels] and the use of checksums for IPv6 tunnels (see Section 3.4.1).

- o Support for operations and maintenance [I-D.ietf-intarea-tunnels].

At the same time, the tunneled traffic is application traffic like any other from the perspective of the networks the tunnel transmits over. This document only touches upon the congestion control considerations for implementing UDP tunnels; a discussion of other required tunneling behavior is out of scope.

3.2. Message Size Guidelines

IP fragmentation lowers the efficiency and reliability of Internet communication. The loss of a single fragment results in the loss of an entire fragmented packet, because even if all other fragments are received correctly, the original packet cannot be reassembled and delivered. This fundamental issue with fragmentation exists for both IPv4 and IPv6.

In addition, some network address translators (NATs) and firewalls drop IP fragments. The network address translation performed by a NAT only operates on complete IP packets, and some firewall policies also require inspection of complete IP packets. Even with these being the case, some NATs and firewalls simply do not implement the necessary reassembly functionality, and instead choose to drop all fragments. Finally, [RFC4963] documents other issues specific to IPv4 fragmentation.

Due to these issues, an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmission Unit (MTU) along the path to the destination. Consequently, an

application SHOULD either use the path MTU information provided by the IP layer or implement Path MTU Discovery (PMTUD) itself [RFC1191][RFC1981][RFC4821] to determine whether the path to a destination will support its desired message size without fragmentation. However, the ICMP messages that enable path MTU discovery are being increasingly filtered by middleboxes (including Firewalls) [RFC4890]. When the path includes a tunnel, some devices acting as a tunnel ingress discard ICMP messages that originate from network devices over which the tunnel passes, preventing these reaching the UDP endpoint.

Packetization Layer Path MTU Discovery (PLPMTUD) [RFC4821] does not rely upon network support for ICMP messages and is therefore considered more robust than standard PMTUD. It is not susceptible to "black holing" of ICMP message. To operate, PLPMTUD requires changes to the way the transport is used, both to transmit probe packets, and to account for the loss or success of these probes. This updates not only the PMTU algorithm, it also impacts loss recovery, congestion control, etc. These updated mechanisms can be implemented within a connection-oriented transport (e.g., TCP, SCTP, DCCP), but are not a part of UDP, but this type of feedback is not typically present for unidirectional applications.

PLPMTUD therefore places additional design requirements on a UDP application that wishes to use this method. This is especially true for UDP tunnels, because the overhead of sending probe packets needs to be accounted for and may require adding a congestion control mechanism to the tunnel (see Section 3.1.11) as well as complicating the data path at a tunnel decapsulator.

Applications that do not follow this recommendation to do PMTU/ PLPMTUD discovery SHOULD still avoid sending UDP datagrams that would result in IP packets that exceed the path MTU. Because the actual path MTU is unknown, such applications SHOULD fall back to sending messages that are shorter than the default effective MTU for sending (EMTU_S in [RFC1122]). For IPv4, EMTU_S is the smaller of 576 bytes and the first-hop MTU [RFC1122]. For IPv6, EMTU_S is 1280 bytes [RFC2460]. The effective PMTU for a directly connected destination (with no routers on the path) is the configured interface MTU, which could be less than the maximum link payload size. Transmission of minimum-sized UDP datagrams is inefficient over paths that support a larger PMTU, which is a second reason to implement PMTU discovery.

To determine an appropriate UDP payload size, applications MUST subtract the size of the IP header (which includes any IPv4 optional headers or IPv6 extension headers) as well as the length of the UDP header (8 bytes) from the PMTU size. This size, known as the Maximum Segment Size (MSS), can be obtained from the TCP/IP stack [RFC1122].

Applications that do not send messages that exceed the effective PMTU of IPv4 or IPv6 need not implement any of the above mechanisms. Note that the presence of tunnels can cause an additional reduction of the effective PMTU [I-D.ietf-intarea-tunnels], so implementing PMTU discovery may be beneficial.

Applications that fragment an application-layer message into multiple UDP datagrams SHOULD perform this fragmentation so that each datagram can be received independently, and be independently retransmitted in the case where an application implements its own reliability mechanisms.

3.3. Reliability Guidelines

Application designers are generally aware that UDP does not provide any reliability, e.g., it does not retransmit any lost packets. Often, this is a main reason to consider UDP as a transport protocol. Applications that do require reliable message delivery MUST implement an appropriate mechanism themselves.

UDP also does not protect against datagram duplication, i.e., an application may receive multiple copies of the same UDP datagram, with some duplicates arriving potentially much later than the first. Application designers SHOULD handle such datagram duplication gracefully, and may consequently need to implement mechanisms to detect duplicates. Even if UDP datagram reception triggers only idempotent operations, applications may want to suppress duplicate datagrams to reduce load.

Applications that require ordered delivery MUST reestablish datagram ordering themselves. The Internet can significantly delay some packets with respect to others, e.g., due to routing transients, intermittent connectivity, or mobility. This can cause reordering, where UDP datagrams arrive at the receiver in an order different from the transmission order.

Applications that use multiple transport ports need to be robust to reordering between sessions. Load-balancing techniques within the network, such as Equal Cost Multipath (ECMP) forwarding can also result in a lack of ordering between different transport sessions, even between the same two network endpoints.

It is important to note that the time by which packets are reordered or after which duplicates can still arrive can be very large. Even more importantly, there is no well-defined upper boundary here. [RFC0793] defines the maximum delay a TCP segment should experience -- the Maximum Segment Lifetime (MSL) -- as 2 minutes. No other RFC defines an MSL for other transport protocols or IP itself. The MSL

value defined for TCP is conservative enough that it SHOULD be used by other protocols, including UDP. Therefore, applications SHOULD be robust to the reception of delayed or duplicate packets that are received within this 2-minute interval.

Retransmission of lost packets or messages is a common reliability mechanism. Such retransmissions can increase network load in response to congestion, worsening that congestion. Any application that uses retransmission is responsible for congestion control of its retransmissions (as well as the application's original traffic), and hence is subject to the Congestion Control guidelines in Section 3.1. Guidance on the appropriate measurement of RTT in Section 3.1.1 also applies for timers used for retransmission packet loss detection.

Instead of implementing these relatively complex reliability mechanisms by itself, an application that requires reliable and ordered message delivery SHOULD whenever possible choose an IETF standard transport protocol that provides these features.

3.4. Checksum Guidelines

The UDP header includes an optional, 16-bit one's complement checksum that provides an integrity check. These checks are not strong from a coding or cryptographic perspective, and are not designed to detect physical-layer errors or malicious modification of the datagram [RFC3819]. Application developers SHOULD implement additional checks where data integrity is important, e.g., through a Cyclic Redundancy Check (CRC) or keyed or non-keyed cryptographic hash included with the data to verify the integrity of an entire object/file sent over the UDP service.

The UDP checksum provides a statistical guarantee that the payload was not corrupted in transit. It also allows the receiver to verify that it was the intended destination of the packet, because it covers the IP addresses, port numbers, and protocol number, and it verifies that the packet is not truncated or padded, because it covers the size field. It therefore protects an application against receiving corrupted payload data in place of, or in addition to, the data that was sent. More description of the set of checks performed using the checksum field is provided in Section 3.1 of [RFC6396].

Applications SHOULD enable UDP checksums [RFC1122]. For IPv4, [RFC0768] permits an option to disable their use, by setting a zero checksum value. An application is permitted to optionally discard UDP datagrams with a zero checksum [RFC1122].

When UDP is used over IPv6, the UDP checksum is relied upon to protect both the IPv6 and UDP headers from corruption (because IPv6

lacks a checksum) and MUST be used as specified in [RFC2460]. Under specific conditions a UDP application is allowed to use a zero UDP zero-checksum mode with a tunnel protocol (see Section 3.4.1).

Applications that choose to disable UDP checksums MUST NOT make assumptions regarding the correctness of received data and MUST behave correctly when a UDP datagram is received that was originally sent to a different destination or is otherwise corrupted.

3.4.1. IPv6 Zero UDP Checksum

[RFC6935] defines a method that enables use of a zero UDP zero-checksum mode with a tunnel protocol, providing that the method satisfies the requirements in [RFC6936]. The application MUST implement mechanisms and/or usage restrictions when enabling this mode. This includes defining the scope for usage and measures to prevent leakage of traffic to other UDP applications (see Appendix A Section 3.6). These additional design requirements for using a zero IPv6 UDP checksum are not present for IPv4, since the IPv4 header validates information that is not protected in an IPv6 packet. Key requirements are:

- o Use of the UDP checksum with IPv6 MUST be the default configuration for all implementations [RFC6935]. The receiving endpoint MUST only allow the use of UDP zero-checksum mode for IPv6 on a UDP destination port that is specifically enabled.

- o An application that support a checksum different to that in [RFC2460] MUST comply with all implementation requirements specified in Section 4 of [RFC6936] and with the usage requirements specified in Section 5 of [RFC6936].

- o A UDP application MUST check that the source and destination IPv6 addresses are valid for any packets with a UDP zero-checksum and MUST discard any packet for which this check fails. To protect from misdelivery, new encapsulation designs SHOULD include an integrity check at the transport layer that includes at least the IPv6 header, the UDP header and the shim header for the encapsulation, if any [RFC6936].

- o One way to help satisfy the requirements of [RFC6936] may be to limit the usage of such tunnels, e.g., to constrain traffic to an operator network, as discussed in Section 3.6. The encapsulation defined for MPLS in UDP [RFC7510] chooses this approach.

As in IPv4, IPv6 applications that choose to disable UDP checksums MUST NOT make assumptions regarding the correctness of received data

and MUST behave correctly when a UDP datagram is received that was originally sent to a different destination or is otherwise corrupted.

IPv6 datagrams with a zero UDP checksum will not be passed by any middlebox that validates the checksum based on [RFC2460] or that updates the UDP checksum field, such as NATs or firewalls. Changing this behavior would require such middleboxes to be updated to correctly handle datagrams with zero UDP checksums To ensure end-to-end robustness, applications that may be deployed in the general Internet MUST provide a mechanism to safely fall back to using a checksum when a path change occurs that redirects a zero UDP checksum flow over a path that includes a middlebox that discards IPv6 datagrams with a zero UDP checksum.

3.4.2. UDP-Lite

A special class of applications can derive benefit from having partially-damaged payloads delivered, rather than discarded, when using paths that include error-prone links. Such applications can tolerate payload corruption and MAY choose to use the Lightweight User Datagram Protocol (UDP-Lite) [RFC3828] variant of UDP instead of basic UDP. Applications that choose to use UDP-Lite instead of UDP should still follow the congestion control and other guidelines described for use with UDP in Section 3.

UDP-Lite changes the semantics of the UDP "payload length" field to that of a "checksum coverage length" field. Otherwise, UDP-Lite is semantically identical to UDP. The interface of UDP-Lite differs from that of UDP by the addition of a single (socket) option that communicates the checksum coverage length: at the sender, this specifies the intended checksum coverage, with the remaining unprotected part of the payload called the "error-insensitive part." By default, the UDP-Lite checksum coverage extends across the entire datagram. If required, an application may dynamically modify this length value, e.g., to offer greater protection to some messages. UDP-Lite always verifies that a packet was delivered to the intended destination, i.e., always verifies the header fields. Errors in the insensitive part will not cause a UDP datagram to be discarded by the destination. Applications using UDP-Lite therefore MUST NOT make assumptions regarding the correctness of the data received in the insensitive part of the UDP-Lite payload.

A UDP-Lite sender SHOULD select the minimum checksum coverage to include all sensitive payload information. For example, applications that use the Real-Time Protocol (RTP) [RFC3550] will likely want to protect the RTP header against corruption. Applications, where appropriate, MUST also introduce their own appropriate validity

checks for protocol information carried in the insensitive part of the UDP-Lite payload (e.g., internal CRCs).

A UDP-Lite receiver MUST set a minimum coverage threshold for incoming packets that is not smaller than the smallest coverage used by the sender [RFC3828]. The receiver SHOULD select a threshold that is sufficiently large to block packets with an inappropriately short coverage field. This may be a fixed value, or may be negotiated by an application. UDP-Lite does not provide mechanisms to negotiate the checksum coverage between the sender and receiver. This therefore needs to be performed by the application.

Applications can still experience packet loss when using UDP-Lite. The enhancements offered by UDP-Lite rely upon a link being able to intercept the UDP-Lite header to correctly identify the partial coverage required. When tunnels and/or encryption are used, this can result in UDP-Lite datagrams being treated the same as UDP datagrams, i.e., result in packet loss. Use of IP fragmentation can also prevent special treatment for UDP-Lite datagrams, and this is another reason why applications SHOULD avoid IP fragmentation (Section 3.2).

UDP-Lite is supported in some endpoint protocol stacks. Current support for middlebox traversal using UDP-Lite is poor, because UDP-Lite uses a different IPv4 protocol number or IPv6 "next header" value than that used for UDP; therefore, few middleboxes are currently able to interpret UDP-Lite and take appropriate actions when forwarding the packet. This makes UDP-Lite less suited for applications needing general Internet support, until such time as UDP-Lite has achieved better support in middleboxes.

3.5. Middlebox Traversal Guidelines

Network address translators (NATs) and firewalls are examples of intermediary devices ("middleboxes") that can exist along an end-to-end path. A middlebox typically performs a function that requires it to maintain per-flow state. For connection-oriented protocols, such as TCP, middleboxes snoop and parse the connection-management information, and create and destroy per-flow state accordingly. For a connectionless protocol such as UDP, this approach is not possible. Consequently, middleboxes can create per-flow state when they see a packet that -- according to some local criteria -- indicates a new flow, and destroy the state after some time during which no packets belonging to the same flow have arrived.

Depending on the specific function that the middlebox performs, this behavior can introduce a time-dependency that restricts the kinds of UDP traffic exchanges that will be successful across the middlebox. For example, NATs and firewalls typically define the partial path on

one side of them to be interior to the domain they serve, whereas the partial path on their other side is defined to be exterior to that domain. Per-flow state is typically created when the first packet crosses from the interior to the exterior, and while the state is present, NATs and firewalls will forward return traffic. Return traffic that arrives after the per-flow state has timed out is dropped, as is other traffic that arrives from the exterior.

Many applications that use UDP for communication operate across middleboxes without needing to employ additional mechanisms. One example is the Domain Name System (DNS), which has a strict request-response communication pattern that typically completes within seconds.

Other applications may experience communication failures when middleboxes destroy the per-flow state associated with an application session during periods when the application does not exchange any UDP traffic. Applications SHOULD be able to gracefully handle such communication failures and implement mechanisms to re-establish application-layer sessions and state.

For some applications, such as media transmissions, this re-synchronization is highly undesirable, because it can cause user-perceivable playback artifacts. Such specialized applications MAY send periodic keep-alive messages to attempt to refresh middlebox state (e.g., [RFC7675]). It is important to note that keep-alive messages are not recommended for general use -- they are unnecessary for many applications and can consume significant amounts of system and network resources.

An application that needs to employ keep-alive messages to deliver useful service over UDP in the presence of middleboxes SHOULD NOT transmit them more frequently than once every 15 seconds and SHOULD use longer intervals when possible. No common timeout has been specified for per-flow UDP state for arbitrary middleboxes. NATs require a state timeout of 2 minutes or longer [RFC4787]. However, empirical evidence suggests that a significant fraction of currently deployed middleboxes unfortunately use shorter timeouts. The timeout of 15 seconds originates with the Interactive Connectivity Establishment (ICE) protocol [RFC5245]. When an application is deployed in a controlled environment, the deployer SHOULD investigate whether the target environment allows applications to use longer intervals, or whether it offers mechanisms to explicitly control middlebox state timeout durations, for example, using the Port Control Protocol (PCP) [RFC6887], Middlebox Communications (MIDCOM) [RFC3303], Next Steps in Signaling (NSIS) [RFC5973], or Universal Plug and Play (UPnP) [UPnP]. It is RECOMMENDED that applications apply slight random variations ("jitter") to the timing of keep-alive

transmissions, to reduce the potential for persistent synchronization between keep-alive transmissions from different hosts [RFC7675].

Sending keep-alive messages is not a substitute for implementing a mechanism to recover from broken sessions. Like all UDP datagrams, keep-alive messages can be delayed or dropped, causing middlebox state to time out. In addition, the congestion control guidelines in Section 3.1 cover all UDP transmissions by an application, including the transmission of middlebox keep-alive messages. Congestion control may thus lead to delays or temporary suspension of keep-alive transmission.

Keep-alive messages are NOT RECOMMENDED for general use. They are unnecessary for many applications and may consume significant resources. For example, on battery-powered devices, if an application needs to maintain connectivity for long periods with little traffic, the frequency at which keep-alive messages are sent can become the determining factor that governs power consumption, depending on the underlying network technology.

Because many middleboxes are designed to require keep-alive messages for TCP connections at a frequency that is much lower than that needed for UDP, this difference alone can often be sufficient to prefer TCP over UDP for these deployments. On the other hand, there is anecdotal evidence that suggests that direct communication through middleboxes, e.g., by using ICE [RFC5245], does succeed less often with TCP than with UDP. The trade-offs between different transport protocols -- especially when it comes to middlebox traversal -- deserve careful analysis.

UDP applications that could be deployed in the Internet need to be designed understanding that there are many variants of middlebox behavior, and although UDP is connectionless, middleboxes often maintain state for each UDP flow. Using multiple UDP flows can consume available state space and also can lead to changes in the way the middlebox handles subsequent packets (either to protect its internal resources, or to prevent perceived misuse). The probability of path failure can increase when applications use multiple UDP flows in parallel (see Section 5.1.2 for recommendations on usage of multiple ports).

3.6. Limited Applicability and Controlled Environments

Two different types of applicability have been identified for the specification of IETF applications that utilize UDP:

General Internet. By default, IETF specifications target deployment on the general Internet. Experience has shown that successful

protocols developed in one specific context or for a particular application tend to become used in a wider range of contexts. For example, a protocol with an initial deployment within a local area network may subsequently be used over a virtual network that traverses the Internet, or in the Internet in general. Applications designed for general Internet use may experience a range of network device behaviors, and in particular should consider whether applications need to operate over paths that may include middleboxes.

Controlled Environment. A protocol/encapsulation/tunnel could be designed to be used only within a controlled environment. For example, an application designed for use by a network operator might only be deployed within the network of that single network operator or on networks of an adjacent set of cooperating network operators. The application traffic may then be managed to avoid congestion, rather than relying on built-in mechanisms, which are required when operating over the general Internet. Applications that target a limited applicability use case may be able to take advantage of specific hardware (e.g., carrier-grade equipment) or underlying protocol features of the subnetwork over which they are used.

Specifications addressing a limited applicability use case or a controlled environment SHOULD identify how in their restricted deployment a level of safety is provided that is equivalent to that of a protocol designed for operation over the general Internet (e.g., a design based on extensive experience with deployments of particular methods that provide features that cannot be expected in general Internet equipment and the robustness of the design of MPLS to corruption of headers both helped justify use of an alternate UDP integrity check [RFC7510]).

An IETF specification targeting a controlled environment is expected to provide an applicability statement that restricts the application traffic to the controlled environment, and would be expected to describe how methods can be provided to discourage or prevent escape of corrupted packets from the environment (for example, section 5 of [RFC7510]).

4. Multicast UDP Usage Guidelines

This section complements Section 3 by providing additional guidelines that are applicable to multicast and broadcast usage of UDP.

Multicast and broadcast transmission [RFC1112] usually employ the UDP transport protocol, although they may be used with other transport protocols (e.g., UDP-Lite).

There are currently two models of multicast delivery: the Any-Source Multicast (ASM) model as defined in [RFC1112] and the Source-Specific Multicast (SSM) model as defined in [RFC4607]. ASM group members will receive all data sent to the group by any source, while SSM constrains the distribution tree to only one single source.

Specialized classes of applications also use UDP for IP multicast or broadcast [RFC0919]. The design of such specialized applications requires expertise that goes beyond simple, unicast-specific guidelines, since these senders may transmit to potentially very many receivers across potentially very heterogeneous paths at the same time, which significantly complicates congestion control, flow control, and reliability mechanisms.

This section provides guidance on multicast and broadcast UDP usage. Use of broadcast by an application is normally constrained by routers to the local subnetwork. However, use of tunneling techniques and proxies can and does result in some broadcast traffic traversing Internet paths. These guidelines therefore also apply to broadcast traffic.

The IETF has defined a reliable multicast framework [RFC3048] and several building blocks to aid the designers of multicast applications, such as [RFC3738] or [RFC4654].

Senders to anycast destinations must be aware that successive messages sent to the same anycast IP address may be delivered to different anycast nodes, i.e., arrive at different locations in the topology.

Most UDP tunnels that carry IP multicast traffic use a tunnel encapsulation with a unicast destination address, such as Automatic Multicast Tunneling [RFC7450]. These MUST follow the same requirements as a tunnel carrying unicast data (see Section 3.1.11). There are deployment cases and solutions where the outer header of a UDP tunnel contains a multicast destination address, such as [RFC6513]. These cases are primarily deployed in controlled environments over reserved capacity, often operating within a single administrative domain, or between two domains over a bi-laterally agreed upon path with reserved capacity, and so congestion control is OPTIONAL, but circuit breaker techniques are still RECOMMENDED in order to restore some degree of service should the offered load exceed the reserved capacity (e.g., due to misconfiguration).

4.1. Multicast Congestion Control Guidelines

Unicast congestion-controlled transport mechanisms are often not applicable to multicast distribution services, or simply do not scale to large multicast trees, since they require bi-directional communication and adapt the sending rate to accommodate the network conditions to a single receiver. In contrast, multicast distribution trees may fan out to massive numbers of receivers, which limits the scalability of an in-band return channel to control the sending rate, and the one-to-many nature of multicast distribution trees prevents adapting the rate to the requirements of an individual receiver. For this reason, generating TCP-compatible aggregate flow rates for Internet multicast data, either native or tunneled, is the responsibility of the application implementing the congestion control.

Applications using multicast SHOULD provide appropriate congestion control. Multicast congestion control needs to be designed using mechanisms that are robust to the potential heterogeneity of both the multicast distribution tree and the receivers belonging to a group. Heterogeneity may manifest itself in some receivers experiencing more loss that others, higher delay, and/or less ability to respond to network conditions. Congestion control is particularly important for any multicast session where all or part of the multicast distribution tree spans an access network (e.g., a home gateway). Two styles of congestion control have been defined in the RFC-series:

- o Feedback-based congestion control, in which the sender receives multicast or unicast UDP messages from the receivers allowing it to assess the level of congestion and then adjust the sender rate(s) (e.g., [RFC5740],[RFC4654]). Multicast methods may operate on longer timescales than for unicast (e.g., due to the higher group RTT of a heterogeneous group). A control method could decide not to reduce the rate of the entire multicast group in response to a control message received from a single receiver (e.g., a sender could set a minimum rate and decide to request a congested receiver to leave the multicast group and could also decide to distribute content to these congested receivers at a lower rate using unicast congestion control).

- o Receiver-driven congestion control, which does not require a receiver to send explicit UDP control messages for congestion control (e.g., [RFC3738], [RFC5775]). Instead, the sender distributes the data across multiple IP multicast groups (e.g., using a set of {S,G} channels). Each receiver determines its own level of congestion and controls its reception rate using only multicast join/leave messages sent in the network control plane. This method scales to arbitrary large groups of receivers.

Any multicast-enabled receiver may attempt to join and receive traffic from any group. This may imply the need for rate limits on individual receivers or the aggregate multicast service. Note there is no way at the transport layer to prevent a join message propagating to the next-hop router.

Some classes of multicast applications support applications that can monitor the user-level quality of the transfer at the receiver. Applications that can detect a significant reduction in user quality SHOULD regard this as a congestion signal (e.g., to leave a group using layered multicast encoding) or, if not, SHOULD use this signal to provide a circuit breaker to terminate the flow by leaving the multicast group.

4.1.1. Bulk Transfer Multicast Applications

Applications that perform bulk transmission of data over a multicast distribution tree, i.e., applications that exchange more than a few UDP datagrams per RTT, SHOULD implement a method for congestion control. The currently RECOMMENDED IETF methods are: Asynchronous Layered Coding (ALC) [RFC5775], TCP-Friendly Multicast Congestion Control (TFMCC) [RFC4654], Wave and Equation Based Rate Control (WEBRC) [RFC3738], NACK-Oriented Reliable Multicast (NORM) transport protocol [RFC5740], File Delivery over Unidirectional Transport (FLUTE) [RFC6726], Real Time Protocol/Control Protocol (RTP/RTCP) [RFC3550].

An application can alternatively implement another congestion control schemes following the guidelines of [RFC2887] and utilizing the framework of [RFC3048]. Bulk transfer applications that choose not to implement [RFC4654], [RFC5775], [RFC3738], [RFC5740], [RFC6726], or [RFC3550] SHOULD implement a congestion control scheme that results in bandwidth use that competes fairly with TCP within an order of magnitude.

Section 2 of [RFC3551] states that multimedia applications SHOULD monitor the packet loss rate to ensure that it is within acceptable parameters. Packet loss is considered acceptable if a TCP flow across the same network path under the same network conditions would achieve an average throughput, measured on a reasonable timescale, that is not less than that of the UDP flow. The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in timescale and throughput.

4.1.2. Low Data-Volume Multicast Applications

All the recommendations in Section 3.1.3 are also applicable to low data-volume multicast applications.

4.2. Message Size Guidelines for Multicast

A multicast application SHOULD NOT send UDP datagrams that result in IP packets that exceed the effective MTU as described in section 3 of [RFC6807]. Consequently, an application SHOULD either use the effective MTU information provided by the Population Count Extensions to Protocol Independent Multicast [RFC6807] or implement path MTU discovery itself (see Section 3.2) to determine whether the path to each destination will support its desired message size without fragmentation.

5. Programming Guidelines

The de facto standard application programming interface (API) for TCP/IP applications is the "sockets" interface [POSIX]. Some platforms also offer applications the ability to directly assemble and transmit IP packets through "raw sockets" or similar facilities. This is a second, more cumbersome method of using UDP. The guidelines in this document cover all such methods through which an application may use UDP. Because the sockets API is by far the most common method, the remainder of this section discusses it in more detail.

Although the sockets API was developed for UNIX in the early 1980s, a wide variety of non-UNIX operating systems also implement it. The sockets API supports both IPv4 and IPv6 [RFC3493]. The UDP sockets API differs from that for TCP in several key ways. Because application programmers are typically more familiar with the TCP sockets API, this section discusses these differences. [STEVENS] provides usage examples of the UDP sockets API.

UDP datagrams may be directly sent and received, without any connection setup. Using the sockets API, applications can receive packets from more than one IP source address on a single UDP socket. Some servers use this to exchange data with more than one remote host through a single UDP socket at the same time. Many applications need to ensure that they receive packets from a particular source address; these applications MUST implement corresponding checks at the application layer or explicitly request that the operating system filter the received packets.

Many operating systems also allow a UDP socket to be connected, i.e., to bind a UDP socket to a specific pair of addresses and ports. This

is similar to the corresponding TCP sockets API functionality. However, for UDP, this is only a local operation that serves to simplify the local send/receive functions and to filter the traffic for the specified addresses and ports. Binding a UDP socket does not establish a connection -- UDP does not notify the remote end when a local UDP socket is bound. Binding a socket also allows configuring options that affect the UDP or IP layers, for example, use of the UDP checksum or the IP Timestamp option. On some stacks, a bound socket also allows an application to be notified when ICMP error messages are received for its transmissions [RFC1122].

If a client/server application executes on a host with more than one IP interface, the application SHOULD send any UDP responses with an IP source address that matches the IP destination address of the UDP datagram that carried the request (see [RFC1122], Section 4.1.3.5). Many middleboxes expect this transmission behavior and drop replies that are sent from a different IP address, as explained in Section 3.5.

A UDP receiver can receive a valid UDP datagram with a zero-length payload. Note that this is different from a return value of zero from a read() socket call, which for TCP indicates the end of the connection.

UDP provides no flow-control, i.e., the sender at any given time does not know whether the receiver is able to handle incoming transmissions. This is another reason why UDP-based applications need to be robust in the presence of packet loss. This loss can also occur within the sending host, when an application sends data faster than the line rate of the outbound network interface. It can also occur at the destination, where receive calls fail to return all the data that was sent when the application issues them too infrequently (i.e., such that the receive buffer overflows). Robust flow control mechanisms are difficult to implement, which is why applications that need this functionality SHOULD consider using a full-featured transport protocol such as TCP.

When an application closes a TCP, SCTP or DCCP socket, the transport protocol on the receiving host is required to maintain TIME-WAIT state. This prevents delayed packets from the closed connection instance from being mistakenly associated with a later connection instance that happens to reuse the same IP address and port pairs. The UDP protocol does not implement such a mechanism. Therefore, UDP-based applications need to be robust to reordering and delay. One application may close a socket or terminate, followed in time by another application receiving on the same port. This later application may then receive packets intended for the first application that were delayed in the network.

5.1. Using UDP Ports

The rules and procedures for the management of the Service Name and Transport Protocol Port Number Registry are specified in [RFC6335]. Recommendations for use of UDP ports are provided in [RFC7605].

A UDP sender SHOULD NOT use a source port value of zero. A source port number that cannot be easily determined from the address or payload type provides protection at the receiver from data injection attacks by off-path devices. A UDP receiver SHOULD NOT bind to port zero.

Applications SHOULD implement receiver port and address checks at the application layer or explicitly request that the operating system filter the received packets to prevent receiving packets with an arbitrary port. This measure is designed to provide additional protection from data injection attacks from an off-path source (where the port values may not be known).

Applications SHOULD provide a check that protects from off-path data injection, avoiding an application receiving packets that were created by an unauthorized third party. TCP stacks commonly use a randomized source port to provide this protection [RFC6056]; UDP applications should follow the same technique. Middleboxes and end systems often make assumptions about the system ports or user ports, hence it is recommended to use randomized ports in the Dynamic and/or Private Port range. Setting a "randomized" source port also provides greater assurance that reported ICMP errors originate from network systems on the path used by a particular flow. Some UDP applications choose to use a predetermined value for the source port (including some multicast applications), these applications need to therefore employ a different technique. Protection from off-path data attacks can also be provided by randomizing the initial value of another protocol field within the datagram payload, and checking the validity of this field at the receiver (e.g., RTP has random initial sequence number and random media timestamp offsets [RFC3550]).

When using multicast, IP routers perform a reverse-path forwarding (RPF) check for each multicast packet. This provides protection from off-path data injection. When a receiver joins a multicast group and filters based on the source address the filter verifies the sender’s IP address. This is always the case when using a SSM {S,G} channel.

5.1.1. Usage of UDP for source port entropy and the IPv6 Flow Label

Some applications use the UDP datagram header as a source of entropy for network devices that implement ECMP [RFC6438]. A UDP tunnel application targeting this usage, encapsulates an inner packet using

UDP, where the UDP source port value forms a part of the entropy that can be used to balance forwarding of network traffic by the devices that use ECMP. A sending tunnel endpoint selects a source port value in the UDP datagram header that is computed from the inner flow information (e.g., the encapsulated packet headers). To provide sufficient entropy the sending tunnel endpoint maps the encapsulated traffic to one of a range of UDP source values. The value SHOULD be within the ephemeral port range, i.e., 49152 to 65535, where the high order two bits of the port are set to one. The available source port entropy of 14 bits (using the ephemeral port range) plus the outer IP addresses seems sufficient for entropy for most ECMP applications [I-D.ietf-rtgwg-dt-encap].

To avoid reordering within an IP flow, the same UDP source port value SHOULD be used for all packets assigned to an encapsulated flow (e.g., using a hash of the relevant headers). The entropy mapping for a flow MAY change over the lifetime of the encapsulated flow [I-D.ietf-rtgwg-dt-encap]. For instance, this could be changed as a Denial of Service (DOS) mitigation, or as a means to effect routing through the ECMP network. However, the source port selected for a flow SHOULD NOT change more than once in every thirty seconds (e.g., as in [I-D.ietf-tsvwg-gre-in-udp-encap]).

The use of the source port field for entropy has several side-effects that need to be considered, including:

- It can increase the probability of misdelivery of corrupted packets, which increases the need for checksum computation or an equivalent mechanism to protect other UDP applications from misdelivery errors Section 3.4.

- It is expected to reduce the probability of successful middlebox traversal Section 3.5. This use of the source port field will often not be suitable for applications targeting deployment in the general Internet.

- It can prevent the field being usable to protect from off-path attacks (described in Section 5.1). Designers therefore need to consider other mechanisms to provide equivalent protection (e.g., to restrict use to a controlled environment [RFC7510] Section 3.6).

The UDP source port number field has also been leveraged to produce entropy with IPv6. However, in the case of IPv6, the "flow label" [RFC6437] may also alternatively be used as entropy for load balancing [RFC6438]. This use of the flow label for load balancing is consistent with the definition of the field, although further clarity was needed to ensure the field can be consistently used for

this purpose. Therefore, an updated IPv6 flow label [RFC6437] and ECMP routing [RFC6438] usage was specified.

To ensure future opportunities to use the flow label, UDP applications SHOULD set the flow label field, even when an entropy value is also set in the source port field (e.g., An IPv6 tunnel endpoint could copy the source port flow entropy value to the IPv6 flow label field [I-D.ietf-tsvwg-gre-in-udp-encap]). Router vendors are encouraged to start using the IPv6 flow label as a part of the flow hash, providing support for IP-level ECMP without requiring use of UDP. The end-to-end use of flow labels for load balancing is a long-term solution. Even if the usage of the flow label has been clarified, there will be a transition time before a significant proportion of endpoints start to assign a good quality flow label to the flows that they originate. The use of load balancing using the transport header fields will likely continue until widespread deployment is finally achieved.

5.1.2. Applications using Multiple UDP Ports

A single application may exchange several types of data. In some cases, this may require multiple UDP flows (e.g., multiple sets of flows, identified by different five-tuples). [RFC6335] recommends application developers not to apply to IANA to be assigned multiple well-known ports (user or system). This does not discuss the implications of using multiple flows with the same well-known port or pairs of dynamic ports (e.g., identified by a service name or signaling protocol).

Use of multiple flows can affect the network in several ways:

- Starting a series of successive connections can increase the number of state bindings in middleboxes (e.g., NAPT or Firewall) along the network path. UDP-based middlebox traversal usually relies on timeouts to remove old state, since middleboxes are unaware when a particular flow ceases to be used by an application.

- Using several flows at the same time may result in seeing different network characteristics for each flow. It cannot be assumed both follow the same path (e.g., when ECMP is used, traffic is intentionally hashed onto different parallel paths based on the port numbers).

- Using several flows can also increase the occupancy of a binding or lookup table in a middlebox (e.g., NAPT or Firewall), which may cause the device to change the way it manages the flow state.

- o Further, using excessive numbers of flows can degrade the ability of a unicast congestion control to react to congestion events, unless the congestion state is shared between all flows in a session. A receiver-driven multicast congestion control requires the sending application to distribute its data over a set of IP multicast groups, each receiver is therefore expected to receive data from a modest number of simultaneously active UDP ports.

Therefore, applications MUST NOT assume consistent behavior of middleboxes when multiple UDP flows are used; many devices respond differently as the number of used ports increases. Using multiple flows with different QoS requirements requires applications to verify that the expected performance is achieved using each individual flow (five-tuple), see Section 3.1.9.

5.2. ICMP Guidelines

Applications can utilize information about ICMP error messages that the UDP layer passes up for a variety of purposes [RFC1122]. Applications SHOULD appropriately validate the payload of ICMP messages to ensure these are received in response to transmitted traffic (i.e., a reported error condition that corresponds to a UDP datagram actually sent by the application). This requires context, such as local state about communication instances to each destination, that although readily available in connection-oriented transport protocols is not always maintained by UDP-based applications. Note that not all platforms have the necessary APIs to support this validation, and some platforms already perform this validation internally before passing ICMP information to the application.

Any application response to ICMP error messages SHOULD be robust to temporary routing failures (sometimes called "soft errors"), e.g., transient ICMP "unreachable" messages ought to not normally cause a communication abort.

ICMP messages are being increasingly filtered by middleboxes. A UDP application therefore SHOULD NOT rely on their delivery for correct and safe operation.

6. Security Considerations

UDP does not provide communications security. Applications that need to protect their communications against eavesdropping, tampering, or message forgery SHOULD employ end-to-end security services provided by other IETF protocols.

UDP applications SHOULD provide protection from off-path data injection attacks using a randomized source port or equivalent technique (see Section 5.1).

Applications that respond to short requests with potentially large responses are a potential vector for amplification attacks, and SHOULD take steps to minimize their potential for being abused as part of a DoS attack. That could mean authenticating the sender before responding; noting that the source IP address of a request is not a useful authenticator, because it can easily be spoofed. Or it may mean otherwise limiting the cases where short unauthenticated requests produce large responses. Applications MAY also want to offer ways to limit the number of requests they respond to in a time interval, in order to cap the bandwidth they consume.

One option for securing UDP communications is with IPsec [RFC4301], which can provide authentication for flows of IP packets through the Authentication Header (AH) [RFC4302] and encryption and/or authentication through the Encapsulating Security Payload (ESP) [RFC4303]. Applications use the Internet Key Exchange (IKE) [RFC7296] to configure IPsec for their sessions. Depending on how IPsec is configured for a flow, it can authenticate or encrypt the UDP headers as well as UDP payloads. If an application only requires authentication, ESP with no encryption but with authentication is often a better option than AH, because ESP can operate across middleboxes. An application that uses IPsec requires the support of an operating system that implements the IPsec protocol suite, and the network path must permit IKE and IPsec traffic. This may become more common with IPv6 deployments [RFC6092].

Although it is possible to use IPsec to secure UDP communications, not all operating systems support IPsec or allow applications to easily configure it for their flows. A second option for securing UDP communications is through Datagram Transport Layer Security (DTLS) [RFC6347][RFC7525]. DTLS provides communication privacy by encrypting UDP payloads. It does not protect the UDP headers. Applications can implement DTLS without relying on support from the operating system.

Many other options for authenticating or encrypting UDP payloads exist. For example, the GSS-API security framework [RFC2743] or Cryptographic Message Syntax (CMS) [RFC5652] could be used to protect UDP payloads. There exist a number of security options for RTP [RFC3550] over UDP, especially to accomplish key-management, see [RFC7201]. These options covers many usages, including point-to-point, centralized group communication as well as multicast. In some applications, a better solution is to protect larger stand-alone objects, such as files or messages, instead of individual UDP

payloads. In these situations, CMS [RFC5652], S/MIME [RFC5751] or OpenPGP [RFC4880] could be used. In addition, there are many non-IETF protocols in this area.

Like congestion control mechanisms, security mechanisms are difficult to design and implement correctly. It is hence RECOMMENDED that applications employ well-known standard security mechanisms such as DTLS or IPsec, rather than inventing their own.

The Generalized TTL Security Mechanism (GTSM) [RFC5082] may be used with UDP applications when the intended endpoint is on the same link as the sender. This lightweight mechanism allows a receiver to filter unwanted packets.

In terms of congestion control, [RFC2309] and [RFC2914] discuss the dangers of congestion-unresponsive flows to the Internet. [I-D.ietf-tsvwg-circuit-breaker] describes methods that can be used to set a performance envelope that can assist in preventing congestion collapse in the absence of congestion control or when the congestion control fails to react to congestion events. This document provides guidelines to designers of UDP-based applications to congestion-control their transmissions, and does not raise any additional security concerns.

Some network operators have experienced surges of UDP attack traffic that are multiple orders of magnitude above the baseline traffic rate for UDP. This can motivate operators to limit the data rate or packet rate of UDP traffic. This may in turn limit the throughput that an application can achieve using UDP and could also result in higher packet loss for UDP traffic that would not be experienced if other transport protocols had been used.

A UDP application with a long-lived association between the sender and receiver, ought to be designed so that the sender periodically checks that the receiver still wants ("consents") to receive traffic and need to be designed to stop if there is no explicit confirmation of this [RFC7675]. Applications that require communications in two directions to implement protocol functions (such as reliability or congestion control) will need to independently check both directions of communication, and may have to exchange keep-alive messages to traverse middleboxes (see Section 3.5).

7. Summary

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

```
+-------------------------------------------------------------+---------+
```

Recommendation	Section
MUST tolerate a wide range of Internet path conditions	3
SHOULD use a full-featured transport (e.g., TCP)	
SHOULD control rate of transmission	3.1
SHOULD perform congestion control over all traffic	
for bulk transfers,	3.1.2
SHOULD consider implementing TFRC	
else, SHOULD in other ways use bandwidth similar to TCP	
for non-bulk transfers,	3.1.3
SHOULD measure RTT and transmit max. 1 datagram/RTT	3.1.1
else, SHOULD send at most 1 datagram every 3 seconds	
SHOULD back-off retransmission timers following loss	
SHOULD provide mechanisms to regulate the bursts of	3.1.6
transmission	
MAY implement ECN; a specific set of application	3.1.7
mechanisms are REQUIRED if ECN is used.	
for DiffServ, SHOULD NOT rely on implementation of PHBs	3.1.8
for QoS-enabled paths, MAY choose not to use CC	3.1.9
SHOULD NOT rely solely on QoS for their capacity	3.1.10
non-CC controlled flows SHOULD implement a transport	
circuit breaker	
MAY implement a circuit breaker for other applications	
for tunnels carrying IP traffic,	3.1.11
SHOULD NOT perform congestion control	
MUST correctly process the IP ECN field	
for non-IP tunnels or rate not determined by traffic,	
SHOULD perform CC or use circuit breaker	3.1.11
SHOULD restrict types of traffic transported by the	
tunnel	
SHOULD NOT send datagrams that exceed the PMTU, i.e.,	3.2
SHOULD discover PMTU or send datagrams < minimum PMTU;	
Specific application mechanisms are REQUIRED if PLPMTUD	
is used.	
SHOULD handle datagram loss, duplication, reordering	3.3
SHOULD be robust to delivery delays up to 2 minutes	

SHOULD enable IPv4 UDP checksum	3.4
SHOULD enable IPv6 UDP checksum; Specific application mechanisms are REQUIRED if a zero IPv6 UDP checksum is used.	3.4.1
SHOULD provide protection from off-path attacks	5.1
else, MAY use UDP-Lite with suitable checksum coverage	3.4.2
SHOULD NOT always send middlebox keep-alive messages MAY use keep-alives when needed (min. interval 15 sec)	3.5
Applications specified for use in limited use (or controlled environments) SHOULD identify equivalent mechanisms and describe their use-case.	3.6
Bulk multicast apps SHOULD implement congestion control	4.1.1
Low volume multicast apps SHOULD implement congestion control	4.1.2
Multicast apps SHOULD use a safe PMTU	4.2
SHOULD avoid using multiple ports MUST check received IP source address	5.1.2
SHOULD validate payload in ICMP messages	5.2
SHOULD use a randomized source port or equivalent technique, and, for client/server applications, SHOULD send responses from source address matching request	6
SHOULD use standard IETF security protocols when needed	6

Table 1: Summary of recommendations

8. IANA Considerations

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

This document raises no IANA considerations.

9. Acknowledgments

The middlebox traversal guidelines in Section 3.5 incorporate ideas from Section 5 of [I-D.ford-behave-app] by Bryan Ford, Pyda Srisuresh, and Dan Kegel. The protocol timer guidelines in Section 3.1.1 were largely contributed by Mark Allman.

G. Fairhurst received funding from the European Union’s Horizon 2020 research and innovation program 2014-2018 under grant agreement No. 644334 (NEAT). Lars Eggert has received funding from the European Union’s Horizon 2020 research and innovation program 2014-2018 under grant agreement No. 644866 (SSICLOPS). This document reflects only the authors’ views and the European Commission is not responsible for any use that may be made of the information it contains.

10. References

10.1. Normative References

[I-D.ietf-tsvwg-circuit-breaker]
Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-15 (work in progress), April 2016.

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

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

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

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <http://www.rfc-editor.org/info/rfc1191>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <http://www.rfc-editor.org/info/rfc2914>.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., Ed., and G. Fairhurst, Ed., "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, DOI 10.17487/RFC3828, July 2004, <http://www.rfc-editor.org/info/rfc3828>.

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

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <http://www.rfc-editor.org/info/rfc4821>.

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <http://www.rfc-editor.org/info/rfc5348>.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

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

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP’s Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <http://www.rfc-editor.org/info/rfc6298>.

10.2. Informative References

[ALLMAN] Allman, M. and E. Blanton, "Notes on burst mitigation for transport protocols", March 2005.

[FABER] Faber, T., Touch, J., and W. Yue, "The TIME-WAIT State in TCP and Its Effect on Busy Servers", Proc. IEEE Infocom, March 1999.

[I-D.ford-behave-app] Ford, B., "Application Design Guidelines for Traversal through Network Address Translators", draft-ford-behave-app-05 (work in progress), March 2007.

[I-D.ietf-aqm-ecn-benefits] Fairhurst, G. and M. Welzl, "The Benefits of using Explicit Congestion Notification (ECN)", draft-ietf-aqm-ecn-benefits-08 (work in progress), November 2015.

[I-D.ietf-avtcore-rtp-circuit-breakers] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", draft-ietf-avtcore-rtp-circuit-breakers-18 (work in progress), August 2016.

[I-D.ietf-intarea-tunnels] Touch, J. and W. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-03 (work in progress), July 2016.

[I-D.ietf-rtgwg-dt-encap] Nordmark, E., Tian, A., Gross, J., Hudson, J., Kreeger, L., Garg, P., Thaler, P., and T. Herbert, "Encapsulation Considerations", draft-ietf-rtgwg-dt-encap-01 (work in progress), March 2016.

[I-D.ietf-tsvwg-gre-in-udp-encap] Yong, L., Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", draft-ietf-tsvwg-gre-in-udp-encap-19 (work in progress), September 2016.

[POSIX] IEEE Std. 1003.1-2001, , "Standard for Information Technology - Portable Operating System Interface (POSIX)", Open Group Technical Standard: Base Specifications Issue 6, ISO/IEC 9945:2002, December 2001.

[RFC0919] Mogul, J., "Broadcasting Internet Datagrams", STD 5, RFC 919, DOI 10.17487/RFC0919, October 1984, <http://www.rfc-editor.org/info/rfc919>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <http://www.rfc-editor.org/info/rfc1112>.

[RFC1536] Kumar, A., Postel, J., Neuman, C., Danzig, P., and S. Miller, "Common DNS Implementation Errors and Suggested Fixes", RFC 1536, DOI 10.17487/RFC1536, October 1993, <http://www.rfc-editor.org/info/rfc1536>.

[RFC1546] Partridge, C., Mendez, T., and W. Milliken, "Host Anycasting Service", RFC 1546, DOI 10.17487/RFC1546, November 1993, <http://www.rfc-editor.org/info/rfc1546>.

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

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <http://www.rfc-editor.org/info/rfc2475>.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <http://www.rfc-editor.org/info/rfc2675>.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <http://www.rfc-editor.org/info/rfc2743>.

[RFC2887] Handley, M., Floyd, S., Whetten, B., Kermode, R., Vicisano, L., and M. Luby, "The Reliable Multicast Design Space for Bulk Data Transfer", RFC 2887, DOI 10.17487/RFC2887, August 2000, <http://www.rfc-editor.org/info/rfc2887>.

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

[RFC3048] Whetten, B., Vicisano, L., Kermode, R., Handley, M., Floyd, S., and M. Luby, "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", RFC 3048, DOI 10.17487/RFC3048, January 2001, <http://www.rfc-editor.org/info/rfc3048>.

[RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, DOI 10.17487/RFC3124, June 2001, <http://www.rfc-editor.org/info/rfc3124>.

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

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

[RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, DOI 10.17487/RFC3303, August 2002, <http://www.rfc-editor.org/info/rfc3303>.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <http://www.rfc-editor.org/info/rfc3493>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <http://www.rfc-editor.org/info/rfc3551>.

[RFC3738] Luby, M. and V. Goyal, "Wave and Equation Based Rate Control (WEBRC) Building Block", RFC 3738, DOI 10.17487/RFC3738, April 2004, <http://www.rfc-editor.org/info/rfc3738>.

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

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <http://www.rfc-editor.org/info/rfc3819>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <http://www.rfc-editor.org/info/rfc4301>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <http://www.rfc-editor.org/info/rfc4302>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

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

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, DOI 10.17487/RFC4341, March 2006, <http://www.rfc-editor.org/info/rfc4341>.

[RFC4342] Floyd, S., Kohler, E., and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, DOI 10.17487/RFC4342, March 2006, <http://www.rfc-editor.org/info/rfc4342>.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <http://www.rfc-editor.org/info/rfc4380>.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <http://www.rfc-editor.org/info/rfc4607>.

[RFC4654] Widmer, J. and M. Handley, "TCP-Friendly Multicast Congestion Control (TFMCC): Protocol Specification", RFC 4654, DOI 10.17487/RFC4654, August 2006, <http://www.rfc-editor.org/info/rfc4654>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <http://www.rfc-editor.org/info/rfc4880>.

[RFC4890] Davies, E. and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls", RFC 4890, DOI 10.17487/RFC4890, May 2007, <http://www.rfc-editor.org/info/rfc4890>.

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

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <http://www.rfc-editor.org/info/rfc4963>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <http://www.rfc-editor.org/info/rfc4987>.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, DOI 10.17487/RFC5082, October 2007, <http://www.rfc-editor.org/info/rfc5082>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <http://www.rfc-editor.org/info/rfc5245>.

[RFC5622] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion ID 4: TCP-Friendly Rate Control for Small Packets (TFRC-SP)", RFC 5622, DOI 10.17487/RFC5622, August 2009, <http://www.rfc-editor.org/info/rfc5622>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

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

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, DOI 10.17487/RFC5740, November 2009, <http://www.rfc-editor.org/info/rfc5740>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <http://www.rfc-editor.org/info/rfc5751>.

[RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, DOI 10.17487/RFC5775, April 2010, <http://www.rfc-editor.org/info/rfc5775>.

[RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", RFC 5971, DOI 10.17487/RFC5971, October 2010, <http://www.rfc-editor.org/info/rfc5971>.

[RFC5973] Stiemerling, M., Tschofenig, H., Aoun, C., and E. Davies, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", RFC 5973, DOI 10.17487/RFC5973, October 2010, <http://www.rfc-editor.org/info/rfc5973>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <http://www.rfc-editor.org/info/rfc6056>.

[RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <http://www.rfc-editor.org/info/rfc6092>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <http://www.rfc-editor.org/info/rfc6335>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC6396] Blunk, L., Karir, M., and C. Labovitz, "Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format", RFC 6396, DOI 10.17487/RFC6396, October 2011, <http://www.rfc-editor.org/info/rfc6396>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <http://www.rfc-editor.org/info/rfc6437>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <http://www.rfc-editor.org/info/rfc6438>.

[RFC6513] Rosen, E., Ed. and R. Aggarwal, Ed., "Multicast in MPLS/BGP IP VPNs", RFC 6513, DOI 10.17487/RFC6513, February 2012, <http://www.rfc-editor.org/info/rfc6513>.

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

[RFC6726] Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport", RFC 6726, DOI 10.17487/RFC6726, November 2012, <http://www.rfc-editor.org/info/rfc6726>.

[RFC6773] Phelan, T., Fairhurst, G., and C. Perkins, "DCCP-UDP: A Datagram Congestion Control Protocol UDP Encapsulation for NAT Traversal", RFC 6773, DOI 10.17487/RFC6773, November 2012, <http://www.rfc-editor.org/info/rfc6773>.

[RFC6807] Farinacci, D., Shepherd, G., Venaas, S., and Y. Cai, "Population Count Extensions to Protocol Independent Multicast (PIM)", RFC 6807, DOI 10.17487/RFC6807, December 2012, <http://www.rfc-editor.org/info/rfc6807>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <http://www.rfc-editor.org/info/rfc6887>.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <http://www.rfc-editor.org/info/rfc6935>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <http://www.rfc-editor.org/info/rfc6936>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <http://www.rfc-editor.org/info/rfc6951>.

[RFC7143] Chadalapaka, M., Satran, J., Meth, K., and D. Black, "Internet Small Computer System Interface (iSCSI) Protocol (Consolidated)", RFC 7143, DOI 10.17487/RFC7143, April 2014, <http://www.rfc-editor.org/info/rfc7143>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

[RFC7450] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, DOI 10.17487/RFC7450, February 2015, <http://www.rfc-editor.org/info/rfc7450>.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <http://www.rfc-editor.org/info/rfc7510>.

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

[RFC7560] Kuehlewind, M., Ed., Scheffenegger, R., and B. Briscoe, "Problem Statement and Requirements for Increased Accuracy in Explicit Congestion Notification (ECN) Feedback", RFC 7560, DOI 10.17487/RFC7560, August 2015, <http://www.rfc-editor.org/info/rfc7560>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

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

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <http://www.rfc-editor.org/info/rfc7657>.

[RFC7675] Perumal, M., Wing, D., Ravindranath, R., Reddy, T., and M. Thomson, "Session Traversal Utilities for NAT (STUN) Usage for Consent Freshness", RFC 7675, DOI 10.17487/RFC7675, October 2015, <http://www.rfc-editor.org/info/rfc7675>.

[STEVENS] Stevens, W., Fenner, B., and A. Rudoff, "UNIX Network Programming, The sockets Networking API", Addison-Wesley, 2004.

[UPnP] UPnP Forum, , "Internet Gateway Device (IGD) Standardized Device Control Protocol V 1.0", November 2001.

Appendix A. Case Study of the Use of IPv6 UDP Zero-Checksum Mode

This appendix provides a brief review of MPLS-in-UDP as an example of a UDP Tunnel Encapsulation that defines a UDP encapsulation. The purpose of the appendix is to provide a concrete example of which mechanisms were required in order to safely use UDP zero-checksum mode for MPLS-in-UDP tunnels over IPv6.

By default, UDP requires a checksum for use with IPv6. An option has been specified that permits a zero IPv6 UDP checksum when used in specific environments, specified in [RFC7510], and defines a set of operational constraints for use of this mode. These are summarized below:

A UDP tunnel or encapsulation using a zero-checksum mode with IPv6 must only be deployed within a single network (with a single network operator) or networks of an adjacent set of co-operating network operators where traffic is managed to avoid congestion, rather than over the Internet where congestion control is required. MPLS-in-UDP has been specified for networks under single administrative control (such as within a single operator’s network) where it is known (perhaps through knowledge of equipment types and lower layer checks) that packet corruption is exceptionally unlikely and where the operator is willing to take the risk of undetected packet corruption.

The tunnel encapsulator SHOULD use different IPv6 addresses for each UDP tunnel that uses the UDP zero-checksum mode, regardless of the decapsulator, to strengthen the decapsulator’s check of the IPv6 source address (i.e., the same IPv6 source address SHOULD NOT be used with more than one IPv6 destination address, independent of whether that destination address is a unicast or multicast address). Use of MPLS-in-UDP may be extended to networks within a set of closely cooperating network administrations (such as network operators who have agreed to work together to jointly provide specific services) [RFC7510].

The requirement for MPLS-in-UDP endpoints to check the source IPv6 address in addition to the destination IPv6 address, plus the strong recommendation against reuse of source IPv6 addresses among MPLS-in-UDP tunnels collectively provide some mitigation for the absence of UDP checksum coverage of the IPv6 header. In addition, the MPLS data plane only forwards packets with valid labels (i.e., labels that have been distributed by the tunnel egress Label Switched Router, LSR), providing some additional opportunity to detect MPLS-in-UDP packet misdelivery when the misdelivered packet contains a label that is not valid for forwarding at the receiving LSR. The expected result for IPv6 UDP zero-checksum mode for MPLS-in-UDP is that corruption of the destination IPv6 address will usually cause packet discard, as

offsetting corruptions to the source IPv6 and/or MPLS top label are unlikely.

Additional assurance is provided by the restrictions in the above exceptions that limit usage of IPv6 UDP zero-checksum mode to well-managed networks for which MPLS packet corruption has not been a problem in practice. Hence, MPLS-in-UDP is suitable for transmission over lower layers in well-managed networks that are allowed by the exceptions stated above and the rate of corruption of the inner IP packet on such networks is not expected to increase by comparison to MPLS traffic that is not encapsulated in UDP. For these reasons, MPLS-in-UDP does not provide an additional integrity check when UDP zero-checksum mode is used with IPv6, and this design is in accordance with requirements 2, 3 and 5 specified in Section 5 of [RFC6936].

The MPLS-in-UDP encapsulation does not provide a mechanism to safely fall back to using a checksum when a path change occurs that redirects a tunnel over a path that includes a middlebox that discards IPv6 datagrams with a zero UDP checksum. In this case, the MPLS-in-UDP tunnel will be black-holed by that middlebox. Recommended changes to allow firewalls, NATs and other middleboxes to support use of an IPv6 zero UDP checksum are described in Section 5 of [RFC6936]. MPLS does not accumulate incorrect state as a consequence of label stack corruption. A corrupt MPLS label results in either packet discard or forwarding (and forgetting) of the packet without accumulation of MPLS protocol state. Active monitoring of MPLS-in-UDP traffic for errors is REQUIRED because the occurrence of errors will result in some accumulation of error information outside the MPLS protocol for operational and management purposes. This design is in accordance with requirement 4 specified in Section 5 of [RFC6936]. In addition, IPv6 traffic with a zero UDP checksum MUST be actively monitored for errors by the network operator.

Operators SHOULD also deploy packet filters to prevent IPv6 packets with a zero UDP checksum from escaping from the network due to misconfiguration or packet errors. In addition, IPv6 traffic with a zero UDP checksum MUST be actively monitored for errors by the network operator.

Appendix B. Revision Notes

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

Changes in draft-ietf-tsvwg-rfc5405bis-19:

- o Addressed IESG review comments.

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

- o Fix a nit.

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

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

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

- o Addressed suggestions by David Black.

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

- o Addressed more suggestions by Takeshi Takahashi.

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

- o Addressed SEC_DIR review by Takeshi Takahashi.
- o Addressed Gen-ART review by Paul Kyzivat.
- o Addressed OPS-DIR review by Tim Chown.
- o Addressed some of Mark Allman’s comments regarding RTTs and RTOs.
- o Addressed some of Brian Trammell;s comments regarding new IETF transports.

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

- o Minor corrections.
- o Changes recommended by Spencer Dawkins.
- o Placed the recommendations on timers within section 3.1
- o Updated the recommendations on reliability to also reference the recommendations on timers.

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

- o Introduced a separate section on the usage of timers to avoid repeating similar guidance in multiple sections.
- o Updated RTT measurement text to align with revised min RTO recommendation for TCP.

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

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

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

- o Address some issues that idnits found.

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

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

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

- o Fix to cross reference in summary table.

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

This update introduces new text in the following sections:

- o The ID from RTGWG on encap Section 7 makes recommendations on entropy. Section 5.1 of the 5405bis draft had a single sentence on use of the UDP source port to inject entropy. Related work such as UDP-in-MPLS and GRE-in-UDP have also made recommendations on entropy usage. A new section has been added to address this.

- o Added reference to RFC2983 on DSCP with tunnels.

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

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

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

- o PLPMTUD section updated.

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

- o Updated summary table

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

This update introduces new text in the following sections:

- o Addressed David Black’s review during WG LC.

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

This update introduces new text in the following sections:

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

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

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

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

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

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

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

- o Added note that this refers to other protocols over IP (E Nordmark, rtg encaps guidance)

- o Added reordering text between sessions (consistent with use of ECMP, rtg encaps guidance)

- o Reworked text on off-path data protection (port usage)

- o Updated summary table

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

- o Added note that guidance may be applicable beyond UDP to abstract (from Erik Nordmark).

- o Small editorial changes to fix English nits.

- o Added a circuit may provide benefit to CC tunnels by controlling envelope.

- o Added tunnels should ingress-filter by packet type (from Erik Nordmark).

- o Added tunnels should perform IETF ECN processing when supporting ECN.

- o Multicast apps may employ CC or a circuit breaker.

- o Added programming guidance on off-path attacks (with C. Perkins).

- o Added reference to ECN benefits.

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

- o Added text on DSCP-usage.

- o More guidance on use of the checksum, including an example of how MPLS/UDP allowed support of a zero IPv6 UDP Checksum in some cases.

- o Added description of diffuse usage.

- o Clarified usage of the source port field.

draft-eggert-tsvwg-rfc5405bis-01 was adopted by the TSVWG and resubmitted as draft-ietf-tsvwg-rfc5405bis-00. There were no technical changes.

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

- o Added Greg Shepherd as a co-author, based on the multicast guidelines that originated with him.

Changes in draft-eggert-tsvwg-rfc5405bis-00 (relative to RFC5405):

- o The words "application designers" were removed from the draft title and the wording of the abstract was clarified abstract.
- o New text to clarify various issues and set new recommendations not previously included in RFC 5405. These include new recommendations for multicast, the use of checksums with IPv6, ECMP, recommendations on port usage, use of ECN, use of DiffServ, circuit breakers (initial text), etc.

Authors' Addresses

Lars Eggert
NetApp
Sonnenallee 1
Kirchheim 85551
Germany

Phone: +49 151 120 55791
EMail: lars@netapp.com
URI: https://eggert.org/

Godred Fairhurst
University of Aberdeen
Department of Engineering
Fraser Noble Building
Aberdeen AB24 3UE
Scotland

EMail: gorry@erg.abdn.ac.uk
URI: http://www.erg.abdn.ac.uk/

Greg Shepherd
Cisco Systems
Tasman Drive
San Jose
USA

EMail: gjshep@gmail.com

Network WG
Internet-Draft
Expires: January 4, 2015
Intended Status: Standards Track
Updates: RFC 2872 (if accepted)

James Polk
Subha Dhesikan
Cisco Systems
July 4, 2014

Resource Reservation Protocol (RSVP) Application-ID Profiles for Voice and Video Streams
draft-ietf-tsvwg-rsvp-app-id-vv-profiles-02

Abstract

RFC 2872 defines an Resource Reservation Protocol (RSVP) object for application identifiers. This document uses that App-ID and gives implementers specific guidelines for differing voice and video stream identifications to nodes along a reservation path, creating specific profiles for voice and video session identification.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

RFC 2872 [RFC2872] describes the usage of policy elements for providing application information in Resource Reservation Protocol (RSVP) signaling [RFC2205]. The intention of providing this information is to enable application-based policy control. However, RFC 2872 does not enumerate any application profiles. The absence of explicit, uniform profiles leads to incompatible handling of these values and misapplied policies. An application profile used by a sender might not be understood by the intermediaries or receiver in a different domain. Therefore, there is a need to enumerate application profiles that are universally understood and applied for correct policy control.

Call control between endpoints has the ability to bind or associate many attributes to a reservation. One new attribute is currently being defined so as to establish the type of traffic contained in that reservation. This is accomplished via assigning a traffic label to the call (or session or flow) [ID-TRAF-CLASS].

This document takes the application traffic classes from [ID-TRAF-CLASS] and places those strings in the APP-ID object defined in RFC 2872. Thus, the intermediary devices (e.g., routers) processing the RSVP message can learn the identified profile within the Application-ID policy element for a particular reservation, and possibly be configured with the profile(s) to understand them

correctly, thus performing the correct admission control.

Another goal of this document is to the ability to signal an application profile which can then be translated into a DSCP value as per the choice of each domain. While the DCLASS object [RFC2996] allows the transfer of DSCP value in an RSVP message, that RFC does not allow the flexibility of having different domains choosing the DSCP value for the traffic classes that they maintain.

How these labels indicate the appropriate Differentiated Services Codepoint (DSCP) is out of scope for this document.

This document will break out each application type and propose how the values in application-id template should be populated for uniformity and interoperability.

1.1 Terminology

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

2. RSVP Application ID Template

The template from RFC 2872 is as follows:

```
             0                   1                   2                   3
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    PE Length (8)              |   P-type = AUTH_APP           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Attribute Length           |   A-type =    |  Sub-type =   |
 |                               | POLICY_LOCATOR|   ASCII_DN    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 Application name as ASCII string              |
 |                         (e.g. SAP.EXE)                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

In line with how this policy element is constructed in RFC 2872, the A-type will remain "POLICY_LOCATOR".

The P-type field is first created in [RFC2752]. This document uses the existing P-type "AUTH_APP" for application traffic class.

The first Sub-type will be mandatory for every profile within this document, and will be "ASCII_DN". No other Sub-types are defined by any profile within this document, but MAY be included by individual implementations - and MUST be ignored if not understood by receiving implementations along the reservation path.

RFC 2872 states the #1 sub-element from RFC 2872 as the "identifier that uniquely identifies the application vendor", which is optional to include. This document modifies this vendor limitation so that the identifier need only be unique - and not limited to an application vendor (identifier). For example, this specification now allows an RFC that defines an industry recognizable term or string to be a valid identifier. For example, a term or string taken from another IETF document, such as "conversational" or "avconf" from [ID-TRAF-CLASS]. This sub-element is still optional to include.

The following subsections will define the values within the above template into specific profiles for voice and video identification.

3. The Voice and Video Application-ID Profiles

This section contains the elements of the Application ID policy object which is used to signal the application classes defined in [ID-TRAF-CLASS].

3.1 The Broadcast Profiles

Broadcast profiles are for minimally buffered one-way streaming flows, such as video surveillance, or Internet based concerts or non-VOD TV broadcasts such as live sporting events.

This document creates Broadcast profiles for

- Broadcast IPTV for audio and video
- Broadcast Live-events for audio and video
- Broadcast Surveillance for audio and video

Here is an example profile for identifying Broadcast Video-Surveillance

```
   AUTH_APP, POLICY_LOCATOR, ASCII_DN,
  "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
   APP=broadcast.video.surveillance, VER="
```

[Editor’s Note: "rfcXXXX" will be replaced with the RFC number assigned to the [ID-TRAF-CLASS] reference. This ’note’ should be removed during the RFC-Editor review process.]

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value at this time.

3.2 The Realtime Interactive Profiles

Realtime Interactive profiles are for on-line gaming, and both remote and virtual avconf applications, in which the timing is particularly important towards the feedback to uses of these applications. This traffic type will generally not be UDP based, with minimal tolerance to RTT delays.

This document creates Realtime Interactive profiles for

- Realtime-Interactive Gaming
- Realtime-Interactive Remote-Desktop
- Realtime-Interactive Virtualized-Desktop

Here is the profile for identifying Realtime-Interactive Gaming

```
   AUTH_APP, POLICY_LOCATOR, ASCII_DN,
  "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
   APP=realtime-interactive.gaming, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.3 The Multimedia Conferencing Profiles

There will be Multimedia Conferencing profiles for presentation data, application sharing and whiteboarding, where these applications will most often be associated with a larger Conversational (audio and/or audio/video) conference. Timing is important, but some minimal delays are acceptable, unlike the case for Realtime-Interactive traffic.

This document creates Multimedia-Conferencing profiles for

- Multimedia-Conferencing presentation-data
- Multimedia-Conferencing presentation-video
- Multimedia-Conferencing presentation-audio
- Multimedia-Conferencing application-sharing
- Multimedia-Conferencing whiteboarding

Here is the profile for identifying Multimedia-Conferencing Application-sharing

```
   AUTH_APP, POLICY_LOCATOR, ASCII_DN,
  "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
   APP=multimedia-conferencing.application-sharing, VER="
```

Where the Globally Unique Identifier (GUID) indicates the RFC reference that created this well-known string [ID-TRAF-CLASS], the

APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.4 The Multimedia Streaming Profiles

Multimedia Streaming profiles are for more significantly buffered one-way streaming flows than Broadcast profiles. These include...

This document creates Multimedia Streaming profiles for

- Multimedia-Streaming multiplex
- Multimedia-Streaming webcast

Here is the profile for identifying Multimedia Streaming webcast

```
   AUTH_APP, POLICY_LOCATOR, ASCII_DN,
  "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
   APP=multimedia-streaming.webcast, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.5 The Conversational Profiles

Conversational category is for realtime bidirectional communications, such as voice or video, and is the most numerous due to the choices of application with or without adjectives. The number of profiles is then doubled because there needs to be one for unadmitted and one for admitted. The IANA section lists all that are currently proposed for registration at this time, therefore there will not be an exhaustive list provided in this section.

This document creates Conversational profiles for

- Conversational Audio
- Conversational Audio Admitted
- Conversational Video
- Conversational Video Admitted
- Conversational Audio Avconf
- Conversational Audio Avconf Admitted
- Conversational Video Avconf
- Conversational Video Avconf Admitted
- Conversational Audio Immersive
- Conversational Audio Immersive Admitted
- Conversational Video Immersive
- Conversational Video Immersive Admitted

Here is an example profile for identifying Conversational Audio:

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.audio, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

4. Security considerations

The security considerations section within RFC 2872 sufficiently covers this document, with one possible exception - someone using the wrong template values (e.g., claiming a reservation is Multimedia Streaming when it is in fact Real-time Interactive). Given that each traffic flow is within separate reservations, and RSVP does not have the ability to police the type of traffic within any reservation, solving for this appears to be administratively handled at best. This is not meant to be a ’punt’, but there really is nothing this template creates that is going to make things any harder for anyone (that we know of now).

5. IANA considerations

5.1 Application Profiles

This document requests IANA create a new registry for the application identification classes similar to the following table within the Resource Reservation Protocol (RSVP) Parameters registry:

Registry Name: RSVP APP-ID Profiles
Reference: [this document]
Registration procedures: Standards Track document [RFC5226]

[Editor’s Note: "rfcXXXX" will be replaced with the RFC number assigned to the [ID-TRAF-CLASS] reference. This ’note’ should be removed during the RFC-Editor review process.]

5.1.1 Broadcast Profiles IANA Registry

Broadcast Audio IPTV Profile

```
P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
                   "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                    APP=broadcast.audio.iptv, VER="
Reference: [this document]
```

```
   Broadcast Video IPTV Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=broadcast.video.iptv, VER="
     Reference: [this document]

   Broadcast Audio Live-events Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=broadcast.audio.live-events, VER="
     Reference: [this document]

   Broadcast Video Live-events Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=broadcast.video.live-events, VER="
     Reference: [this document]

   Broadcast Audio-Surveillance Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=broadcast.audio.surveillance, VER="
     Reference: [this document]

   Broadcast Video-Surveillance Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=broadcast.video.surveillance, VER="
     Reference: [this document]
```

5.1.2 Realtime-Interactive Profiles IANA Registry

```
   Realtime-Interactive Gaming Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
```

```
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP= realtime-interactive.gaming, VER="
     Reference: [this document]

   Real-time Interactive Remote-Desktop Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=realtime-interactive.remote-desktop, VER="
     Reference: [this document]

   Real-time Interactive Virtualized-Desktop Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=realtime-interactive.
                            remote-desktop.virtual, VER="
     Reference: [this document]

   Real-time Interactive Telemetry Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=realtime-interactive.telemetry, VER="
     Reference: [this document]
```

5.1.3 Multimedia-Conferencing Profiles IANA Registry

```
   Multimedia-Conferencing Presentation-Data Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                      "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                       APP= multimedia-conferencing.presentation-data,
                       VER="
     Reference: [this document]

   Multimedia-Conferencing Presentation-Video Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                      "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
```

```
                        APP= multimedia-conferencing.presentation-video,
                        VER="
     Reference: [this document]

   Multimedia-Conferencing Presentation-Audio Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                      "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                       APP= multimedia-conferencing.presentation-audio,
                       VER="
     Reference: [this document]

   Multimedia-Conferencing Application-Sharing Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                     "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                      APP= multimedia-conferencing.application-sharing,
                      VER="
     Reference: [this document]

   Multimedia-Conferencing Whiteboarding Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                     "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                      APP= multimedia-conferencing.whiteboarding, VER="
     Reference: [this document]
```

5.1.4 Multimedia-Streaming Profiles IANA Registry

```
   Multimedia-Streaming Multiplex Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=multimedia-streaming.multiplex, VER="
     Reference: [this document]

   Multimedia-Streaming Webcast Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=multimedia-streaming.webcast, VER="
```

```
     Reference: [this document]
```

5.1.5 Conversational Profiles IANA Registry

```
   Conversational Audio Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.audio, VER="
     Reference: [this document]

   Conversational Audio Admitted Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.audio.aq:admitted, VER="
     Reference: [this document]

   Conversational Video Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.video, VER="
     Reference: [this document]

   Conversational Video Admitted Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.video.aq:admitted, VER="
     Reference: [this document]

   Conversational Audio Avconf Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.audio.avconf, VER="
     Reference: [this document]

   Conversational Audio Avconf Admitted Profile
     P-type = AUTH_APP
```

```
A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                     APP=conversational.audio.avconf.aq:admitted,
                     VER="
  Reference: [this document]

Conversational Video Avconf Profile
  P-type = AUTH_APP
  A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                     APP=conversational.video.avconf, VER="
  Reference: [this document]

Conversational Video Avconf Admitted Profile
  P-type = AUTH_APP
  A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                     APP=conversational.video.avconf.aq:admitted,
                     VER="
  Reference: [this document]

Conversational Audio Immersive Profile
  P-type = AUTH_APP
  A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                     APP=conversational.audio.immersive, VER="
  Reference: [this document]

Conversational Audio Immersive Admitted Profile
  P-type = AUTH_APP
  A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                     APP=conversational.audio.immersive.aq:admitted,
                     VER="
  Reference: [this document]

Conversational Video Immersive Profile
  P-type = AUTH_APP
  A-type = POLICY_LOCATOR
  Sub-type = ASCII_DN
  Conformant policy locator =
                    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
```

```
                        APP=conversational.video.immersive, VER="
     Reference: [this document]

   Conversational Video Immersive Admitted Profile
     P-type = AUTH_APP
     A-type = POLICY_LOCATOR
     Sub-type = ASCII_DN
     Conformant policy locator =
                       "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
                        APP=conversational.video.immersive.aq:admitted,
                        VER="
     Reference: [this document]
```

6. Acknowledgments

To Francois Le Faucheur, Paul Jones, Ken Carlberg, Georgios Karagiannis and Glen Lavers for their helpful comments, document reviews and encouragement.

7. References

7.1. Normative References

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997
Functional Specification", RFC 2205, September 1997

[RFC2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997

[RFC2474] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers ", RFC 2474, December 1998

[RFC2750] S. Herzog, "RSVP Extensions for Policy Control", RFC 2750, January 2000

[RFC2872] Y. Bernet, R. Pabbati, "Application and Sub Application Identity Policy Element for Use with RSVP", RFC 2872, June 2000

[RFC2996] Y. Bernet, "Format of the RSVP DCLASS Object", RFC 2996, November 2000

[RFC3182] S. Yadav, R. Yavatkar, R. Pabbati, P. Ford, T. Moore, S. Herzog, R. Hess, "Identity Representation for RSVP", RFC 3182, October 2001

[RFC5226] T. Narten, H. Alvestrand, "Guidelines for Writing an IANA

Considerations Section in RFCs", RFC 5226, May 2008

[ID-TRAF-CLASS] J. Polk, S. Dhesikan, P. Jones, "The Session Description Protocol (SDP) 'trafficclass' Attribute", work in progress, Feb 2013

7.2. Informative References

[RFC4594] J. Babiarz, K. Chan, F Baker, "Configuration Guidelines for Diffserv Service Classes", RFC 4594, August 2006

Authors' Addresses

James Polk
3913 Treemont Circle
Colleyville, Texas, USA
+1.817.271.3552

mailto: jmpolk@cisco.com

Subha Dhesikan
170 W Tasman St
San Jose, CA, USA
+1.408-902-3351

mailto: sdhesika@cisco.com

Appendix - Changes to ID

[Editor's Note: this appendix should be removed in the RFC-Editor's process.]

A.1 - Changes from WG version -00 to WG version -01

The following changes were made in this version:

- corrected nits

- globally replaced GUID link from the MMUSIC Trafficclass ID to the future RFC of that document.

- added profiles for presentation-video and presentation-audio

A.2 - Changes from Individual -04 to WG version -00

The following changes were made in this version:

- changed P-Type from APP_TC back to AUTH_APP, which is already defined.

- fixed nits and inconsistencies

A.3 - Changes from Individual -03 to -04

The following changes were made in this version:

- clarified security considerations section to mean RSVP cannot police the type of traffic within a reservation to know if a traffic flow should be using a different profile, as defined in this document.

- changed existing informative language regarding "... other Sub-types ..." from 'can' to normative 'MAY'.
- editorial changes to clear up minor mistakes

A.4 - Changes from Individual -02 to -03

The following changes were made in this version:

- Added [ID-TRAF-CLASS] as a reference

- Changed to a new format of the profile string.

- Added many new profiles based on the new format into each parent category of Section 3.

- changed the GUID to refer to draft-ietf-mmusic-traffic-class-for-sdp-03.txt

- changed 'desktop' adjective to 'avconf' to keep in alignment with [ID-TRAF-CLASS]

- Have a complete IANA Registry proposal for each application-ID discussed in this draft.

- General text clean-up of the draft.

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 20, 2017

P. Jones
S. Dhesikan
C. Jennings
Cisco Systems
D. Druta
AT&T

August 19, 2016

DSCP Packet Markings for WebRTC QoS
draft-ietf-tsvwg-rtcweb-qos-18

Abstract

Many networks, such as service provider and enterprise networks, can provide different forwarding treatments for individual packets based on Differentiated Services Code Point (DSCP) values on a per-hop basis. This document provides the recommended DSCP values for web browsers to use for various classes of WebRTC traffic.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Differentiated Services Code Point (DSCP) [RFC2474] packet marking can help provide QoS in some environments. This specification provides default packet marking for browsers that support WebRTC applications, but does not change any advice or requirements in other IETF RFCs. The contents of this specification are intended to be a simple set of implementation recommendations based on the previous RFCs.

Networks where these DSCP markings are beneficial (likely to improve QoS for WebRTC traffic) include:

1. Private, wide-area networks. Network administrators have control over remarking packets and treatment of packets.

2. Residential Networks. If the congested link is the broadband uplink in a cable or DSL scenario, often residential routers/NAT support preferential treatment based on DSCP.

3. Wireless Networks. If the congested link is a local wireless network, marking may help.

There are cases where these DSCP markings do not help, but, aside from possible priority inversion for "less than best effort traffic"

(see Section 5), they seldom make things worse if packets are marked appropriately.

DSCP values are in principle site specific, with each site selecting its own code points for controlling per-hop-behavior to influence the QoS for transport-layer flows. However in the WebRTC use cases, the browsers need to set them to something when there is no site specific information. This document describes a subset of DSCP code point values drawn from existing RFCs and common usage for use with WebRTC applications. These code points are intended to be the default values used by a WebRTC application. While other values could be used, using a non-default value may result in unexpected per-hop behavior. It is RECOMMENDED that WebRTC applications use non-default values only in private networks that are configured to use different values.

This specification defines inputs that are provided by the WebRTC application hosted in the browser that aid the browser in determining how to set the various packet markings. The specification also defines the mapping from abstract QoS policies (flow type, priority level) to those packet markings.

2. Terminology

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

The terms "browser" and "non-browser" are defined in [RFC7742] and carry the same meaning in this document.

3. Relation to Other Specifications

This document is a complement to [RFC7657], which describes the interaction between DSCP and real-time communications. That RFC covers the implications of using various DSCP values, particularly focusing on Real-time Transport Protocol (RTP) [RFC3550] streams that are multiplexed onto a single transport-layer flow.

There are a number of guidelines specified in [RFC7657] that apply to marking traffic sent by WebRTC applications, as it is common for multiple RTP streams to be multiplexed on the same transport-layer flow. Generally, the RTP streams would be marked with a value as appropriate from Table 1. A WebRTC application might also multiplex data channel [I-D.ietf-rtcweb-data-channel] traffic over the same 5-tuple as RTP streams, which would also be marked as per that table. The guidance in [RFC7657] says that all data channel traffic would be marked with a single value that is typically different than the

value(s) used for RTP streams multiplexed with the data channel traffic over the same 5-tuple, assuming RTP streams are marked with a value other than default forwarding (DF). This is expanded upon further in the next section.

This specification does not change or override the advice in any other IETF RFCs about setting packet markings. Rather, it simply selects a subset of DSCP values that is relevant in the WebRTC context.

The DSCP value set by the endpoint is not trusted by the network. In addition, the DSCP value may be remarked at any place in the network for a variety of reasons to any other DSCP value, including default forwarding (DF) value to provide basic best effort service. Even so, there is benefit in marking traffic even if it only benefits the first few hops. The implications are discussed in Secton 3.2 of [RFC7657]. Further, a mitigation for such action is through an authorization mechanism. Such an authorization mechanism is outside the scope of this document.

4. Inputs

WebRTC applications send and receive two types of flows of significance to this document:

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

- o data flows which are data channels [I-D.ietf-rtcweb-data-channel]

Each of the RTP streams and distinct data channels consists of all of the packets associated with an independent media entity, so an RTP stream or distinct data channel is not always equivalent to a transport-layer flow defined by a 5-tuple (source address, destination address, source port, destination port, and protocol). There may be multiple RTP streams and data channels multiplexed over the same 5-tuple, with each having a different level of importance to the application and, therefore, potentially marked using different DSCP values than another RTP stream or data channel within the same transport-layer flow. (Note that there are restrictions with respect to marking different data channels carried within the same SCTP association as outlined in Section 5.)

The following are the inputs provided by the WebRTC application to the browser:

- o Flow Type: The application provides this input because it knows if the flow is audio, interactive video [RFC4594] [G.1010] with or without audio, or data.

- o Application Priority: Another input is the relative importance of an RTP stream or data channel. Many applications have multiple flows of the same Flow Type and often some flows are more important than others. For example, in a video conference where there are usually audio and video flows, the audio flow may be more important than the video flow. JavaScript applications can tell the browser whether a particular flow is high, medium, low or very low importance to the application.

[I-D.ietf-rtcweb-transports] defines in more detail what an individual flow is within the WebRTC context and priorities for media and data flows.

Currently in WebRTC, media sent over RTP is assumed to be interactive [I-D.ietf-rtcweb-transports] and browser APIs do not exist to allow an application to to differentiate between interactive and non-interactive video.

5. DSCP Mappings

The DSCP values for each flow type of interest to WebRTC based on application priority are shown in Table 1. These values are based on the framework and recommended values in [RFC4594]. A web browser SHOULD use these values to mark the appropriate media packets. More information on EF can be found in [RFC3246]. More information on AF can be found in [RFC2597]. DF is default forwarding which provides the basic best effort service [RFC2474].

WebRTC use of multiple DSCP values may encounter network blocking of packets with certain DSCP values. See section 4.2 of [I-D.ietf-rtcweb-transports] for further discussion, including how WebRTC implementations establish and maintain connectivity when such blocking is encountered.

Flow Type	Very Low	Low	Medium	High
Audio	CS1 (8)	DF (0)	EF (46)	EF (46)
Interactive Video with or without Audio	CS1 (8)	DF (0)	AF42, AF43 (36, 38)	AF41, AF42 (34, 36)
Non-Interactive Video with or without Audio	CS1 (8)	DF (0)	AF32, AF33 (28, 30)	AF31, AF32 (26, 28)
Data	CS1 (8)	DF (0)	AF11	AF21

Table 1: Recommended DSCP Values for WebRTC Applications

The application priority, indicated by the columns "very low", "low", "Medium", and "high", signifies the relative importance of the flow within the application. It is an input that the browser receives to assist in selecting the DSCP value and adjusting the network transport behavior.

The above table assumes that packets marked with CS1 are treated as "less than best effort", such as the LE behavior described in [RFC3662]. However, the treatment of CS1 is implementation dependent. If an implementation treats CS1 as other than "less than best effort", then the actual priority (or, more precisely, the per-hop-behavior) of the packets may be changed from what is intended. It is common for CS1 to be treated the same as DF, so applications and browsers using CS1 cannot assume that CS1 will be treated differently than DF [RFC7657]. However, it is also possible per [RFC2474] for CS1 traffic to be given better treatment than DF, thus caution should be exercised when electing to use CS1. This is one of the cases where marking packets using these recommendations can make things worse.

Implementers should also note that excess EF traffic is dropped. This could mean that a packet marked as EF may not get through, although the same packet marked with a different DSCP value would have gotten through. This is not a flaw, but how excess EF traffic is intended to be treated.

The browser SHOULD first select the flow type of the flow. Within the flow type, the relative importance of the flow SHOULD be used to select the appropriate DSCP value.

Currently, all WebRTC video is assumed to be interactive [I-D.ietf-rtcweb-transports], for which the Interactive Video DSCP values in Table 1 SHOULD be used. Browsers MUST NOT use the AF3x DSCP values (for Non-Interactive Video in Table 1) for WebRTC applications. Non-browser implementations of WebRTC MAY use the AF3x DSCP values for video that is known not to be interactive, e.g., all video in a WebRTC video playback application that is not implemented in a browser.

The combination of flow type and application priority provides specificity and helps in selecting the right DSCP value for the flow. All packets within a flow SHOULD have the same application priority. In some cases, the selected application priority cell may have multiple DSCP values, such as AF41 and AF42. These offer different drop precedences. The different drop precedence values provides additional granularity in classifying packets within a flow. For example, in a video conference the video flow may have medium application priority, thus either AF42 or AF43 may be selected. More important video packets (e.g., a video picture or frame encoded without any dependency on any prior pictures or frames) might be marked with AF42 and less important packets (e.g., a video picture or frame encoded based on the content of one or more prior pictures or frames) might be marked with AF43 (e.g., receipt of the more important packets enables a video renderer to continue after one or more packets are lost).

It is worth noting that the application priority is utilized by the coupled congestion control mechanism for media flows per [I-D.ietf-rmcat-coupled-cc] and the SCTP scheduler for data channel traffic per [I-D.ietf-rtcweb-data-channel].

For reasons discussed in Section 6 of [RFC7657], if multiple flows are multiplexed using a reliable transport (e.g., TCP) then all of the packets for all flows multiplexed over that transport-layer flow MUST be marked using the same DSCP value. Likewise, all WebRTC data channel packets transmitted over an SCTP association MUST be marked using the same DSCP value, regardless of how many data channels (streams) exist or what kind of traffic is carried over the various SCTP streams. In the event that the browser wishes to change the DSCP value in use for an SCTP association, it MUST reset the SCTP congestion controller after changing values. Frequent changes in the DSCP value used for an SCTP association are discouraged, though, as this would defeat any attempts at effectively managing congestion. It should also be noted that any change in DSCP value that results in a reset of the congestion controller puts the SCTP association back into slow start, which may have undesirable effects on application performance.

For the data channel traffic multiplexed over an SCTP association, it is RECOMMENDED that the DSCP value selected be the one associated with the highest priority requested for all data channels multiplexed over the SCTP association. Likewise, when multiplexing multiple flows over a TCP connection, the DCSP value selected should be the one associated with the highest priority requested for all multiplexed flows.

If a packet enters a network that has no support for a flow type-application priority combination specified in Table 1, then the network node at the edge will remark the DSCP value based on policies. This could result in the flow not getting the network treatment it expects based on the original DSCP value in the packet. Subsequently, if the packet enters a network that supports a larger number of these combinations, there may not be sufficient information in the packet to restore the original markings. Mechanisms for restoring such original DSCP is outside the scope of this document.

In summary, DSCP marking provides neither guarantees nor promised levels of service. However, DSCP marking is expected to provide a statistical improvement in real-time service as a whole. The service provided to a packet is dependent upon the network design along the path, as well as the network conditions at every hop.

6. Security Considerations

Since the JavaScript application specifies the flow type and application priority that determine the media flow DSCP values used by the browser, the browser could consider application use of a large number of higher priority flows to be suspicious. If the server hosting the JavaScript application is compromised, many browsers within the network might simultaneously transmit flows with the same DSCP marking. The DiffServ architecture requires ingress traffic conditioning for reasons that include protecting the network from this sort of attack.

Otherwise, this specification does not add any additional security implications beyond those addressed in the following DSCP-related specifications. For security implications on use of DSCP, please refer to Section 7 of [RFC7657] and Section 6 of [RFC4594]. Please also see [I-D.ietf-rtcweb-security] as an additional reference.

7. IANA Considerations

This specification does not require any actions from IANA.

8. Downward References

This specification contains a downwards reference to [RFC4594] and [RFC7657]. However, the parts of the former RFC used by this specification are sufficiently stable for this downward reference. The guidance in the latter RFC is necessary to understand the Diffserv technology used in this document and the motivation for the recommended DSCP values and procedures.

9. Acknowledgements

Thanks to David Black, Magnus Westerlund, Paolo Severini, Jim Hasselbrook, Joe Marcus, Erik Nordmark, Michael Tuexen, and Brian Carpenter for their invaluable input.

10. Dedication

This document is dedicated to the memory of James Polk, a long-time friend and colleague. James made important contributions to this specification, including serving initially as one of the primary authors. The IETF global community mourns his loss and he will be missed dearly.

11. Document History

Note to RFC Editor: Please remove this section.

This document was originally an individual submission in RTCWeb WG. The RTCWeb working group selected it to be become a WG document. Later the transport ADs requested that this be moved to the TSVWG WG as that seemed to be a better match.

12. References

12.1. Normative References

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ietf-rtcweb-rtp-usage]
Perkins, D., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-26 (work in progress), March 2016.

[I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", draft-ietf-rtcweb-security-08 (work in progress), February 2015.

[I-D.ietf-rtcweb-transports]
Alvestrand, H., "Transports for WebRTC", draft-ietf-rtcweb-transports-15 (work in progress), August 2016.

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

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <http://www.rfc-editor.org/info/rfc4594>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <http://www.rfc-editor.org/info/rfc7657>.

[RFC7742] Roach, A., "WebRTC Video Processing and Codec Requirements", RFC 7742, DOI 10.17487/RFC7742, March 2016, <http://www.rfc-editor.org/info/rfc7742>.

12.2. Informative References

[G.1010] International Telecommunications Union, "End-user multimedia QoS categories", Recommendation ITU-T G.1010, November 2001.

[I-D.ietf-rmcat-coupled-cc]
Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-03 (work in progress), July 2016.

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

[RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, DOI 10.17487/RFC2597, June 1999, <http://www.rfc-editor.org/info/rfc2597>.

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

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services", RFC 3662, DOI 10.17487/RFC3662, December 2003, <http://www.rfc-editor.org/info/rfc3662>.

Authors' Addresses

Paul E. Jones
Cisco Systems

Email: paulej@packetizer.com

Subha Dhesikan
Cisco Systems

Email: sdhesika@cisco.com

Cullen Jennings
Cisco Systems

Email: fluffy@cisco.com

Dan Druta
AT&T

Email: dd5826@att.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 28, 2015

M. Tuexen
Muenster Univ. of Appl. Sciences
R. Stewart
Netflix, Inc.
R. Jesup
WorldGate Communications
S. Loreto
Ericsson

January 24, 2015

DTLS Encapsulation of SCTP Packets
draft-ietf-tsvwg-sctp-dtls-encaps-09.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a transport protocol originally defined to run on top of the network protocols IPv4 or IPv6. This document specifies how SCTP can be used on top of the Datagram Transport Layer Security (DTLS) protocol. Using the encapsulation method described in this document, SCTP is unaware of the protocols being used below DTLS; hence explicit IP addresses cannot be used in the SCTP control chunks. As a consequence, the SCTP associations carried over DTLS can only be single homed.

Status of This Memo

Copyright Notice

Table of Contents

1. Overview

The Stream Control Transmission Protocol (SCTP) as defined in [RFC4960] is a transport protocol running on top of the network protocols IPv4 [RFC0791] or IPv6 [RFC2460]. This document specifies how SCTP is used on top of the Datagram Transport Layer Security (DTLS) protocol. DTLS 1.0 is defined in [RFC4347] and the latest version when this RFC was published, DTLS 1.2, is defined in [RFC6347]. This encapsulation is used for example within the WebRTC protocol suite (see [I-D.ietf-rtcweb-overview] for an overview) for transporting non-SRTP data between browsers. The architecture of this stack is described in [I-D.ietf-rtcweb-data-channel].

[NOTE to RFC-Editor:

Please ensure that the authors double check the above statement about DTLS 1.2 during AUTH48 and then remove this note before publication.

]

```
+----------+
|   SCTP   |
+----------+
|   DTLS   |
+----------+
| ICE/UDP  |
+----------+
```

Figure 1: Basic stack diagram

This encapsulation of SCTP over DTLS over UDP or ICE/UDP (see [RFC5245]) can provide a NAT traversal solution in addition to confidentiality, source authentication, and integrity protected transfers. Please note that using ICE does not necessarily imply that a different packet format is used on the wire.

Please note that the procedures defined in [RFC6951] for dealing with the UDP port numbers do not apply here. When using the encapsulation defined in this document, SCTP is unaware about the protocols used below DTLS.

2. Conventions

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

3. Encapsulation and Decapsulation Procedure

When an SCTP packet is provided to the DTLS layer, the complete SCTP packet, consisting of the SCTP common header and a number of SCTP chunks, is handled as the payload of the application layer protocol of DTLS. When the DTLS layer has processed a DTLS record containing a message of the application layer protocol, the payload is passed to the SCTP layer. The SCTP layer expects an SCTP common header followed by a number of SCTP chunks.

4. General Considerations

An implementation of SCTP over DTLS MUST implement and use a path maximum transmission unit (MTU) discovery method that functions without ICMP to provide SCTP/DTLS with an MTU estimate. An implementation of "Packetization Layer Path MTU Discovery" [RFC4821] either in SCTP or DTLS is RECOMMENDED.

The path MTU discovery is performed by SCTP when SCTP over DTLS is used for data channels (see Section 5 of [I-D.ietf-rtcweb-data-channel]).

5. DTLS Considerations

The DTLS implementation MUST support DTLS 1.0 [RFC4347] and SHOULD support the most recently published version of DTLS, which was DTLS 1.2 [RFC6347] when this RFC was published. In the absence of a revision to this document, the latter requirement applies to all future versions of DTLS when they are published as RFCs. This document will only be revised if a revision to DTLS or SCTP makes a revision to the encapsulation necessary.

[NOTE to RFC-Editor:

Please ensure that the authors double check the above statement about DTLS 1.2 during AUTH48 and then remove this note before publication.

]

SCTP performs segmentation and reassembly based on the path MTU. Therefore the DTLS layer MUST NOT use any compression algorithm.

The DTLS MUST support sending messages larger than the current path MTU. This might result in sending IP level fragmented messages.

If path MTU discovery is performed by the DTLS layer, the method described in [RFC4821] MUST be used. For probe packets, the extension defined in [RFC6520] MUST be used.

If path MTU discovery is performed by the SCTP layer and IPv4 is used as the network layer protocol, the DTLS implementation SHOULD allow the DTLS user to enforce that the corresponding IPv4 packet is sent with the Don’t Fragment (DF) bit set. If controlling the DF bit is not possible, for example due to implementation restrictions, a safe value for the path MTU has to be used by the SCTP stack. It is RECOMMENDED that the safe value does not exceed 1200 bytes. Please note that [RFC1122] only requires end hosts to be able to reassemble fragmented IP packets up to 576 bytes in length.

The DTLS implementation SHOULD allow the DTLS user to set the Differentiated services code point (DSCP) used for IP packets being sent (see [RFC2474]). This requires the DTLS implementation to pass the value through and the lower layer to allow setting this value. If the lower layer does not support setting the DSCP, then the DTLS user will end up with the default value used by protocol stack. Please note that only a single DSCP value can be used for all packets belonging to the same SCTP association.

Using explicit congestion notifications (ECN) in SCTP requires the DTLS layer to pass the ECN bits through and its lower layer to expose access to them for sent and received packets (see [RFC3168]). The implementation of DTLS and its lower layer have to provide this support. If this is not possible, for example due to implementation restrictions, ECN can't be used by SCTP.

6. SCTP Considerations

This section describes the usage of the base protocol and the applicability of various SCTP extensions.

6.1. Base Protocol

This document uses SCTP [RFC4960] with the following restrictions, which are required to reflect that the lower layer is DTLS instead of IPv4 and IPv6 and that SCTP does not deal with the IP addresses or the transport protocol used below DTLS:

- o A DTLS connection MUST be established before an SCTP association can be set up.

- o Multiple SCTP associations MAY be multiplexed over a single DTLS connection. The SCTP port numbers are used for multiplexing and demultiplexing the SCTP associations carried over a single DTLS connection.

- o All SCTP associations are single-homed, because DTLS does not expose any address management to its upper layer. Therefore it is RECOMMENDED to set the SCTP parameter path.max.retrans to association.max.retrans.

- o The INIT and INIT-ACK chunk MUST NOT contain any IPv4 Address or IPv6 Address parameters. The INIT chunk MUST NOT contain the Supported Address Types parameter.

- o The implementation MUST NOT rely on processing ICMP or ICMPv6 packets, since the SCTP layer most likely is unable to access the SCTP common header in the plain text of the packet, which triggered the sending of the ICMP or ICMPv6 packet. This applies in particular to path MTU discovery when performed by SCTP.

- o If the SCTP layer is notified about a path change by its lower layers, SCTP SHOULD retest the Path MTU and reset the congestion state to the initial state. The window-based congestion control method specified in [RFC4960], resets the congestion window and slow start threshold to their initial values.

6.2. Padding Extension

When the SCTP layer performs path MTU discovery as specified in [RFC4821], the padding extension defined in [RFC4820] MUST be supported and used for probe packets (HEARTBEAT chunks bundled with PADDING chunks [RFC4820]).

6.3. Dynamic Address Reconfiguration Extension

If the dynamic address reconfiguration extension defined in [RFC5061] is used, ASCONF chunks MUST use wildcard addresses only.

6.4. SCTP Authentication Extension

The SCTP authentication extension defined in [RFC4895] can be used with DTLS encapsulation, but does not provide any additional benefit.

6.5. Partial Reliability Extension

Partial reliability as defined in [RFC3758] can be used in combination with DTLS encapsulation. It is also possible to use additional PR-SCTP policies, for example the ones defined in [I-D.ietf-tsvwg-sctp-prpolicies].

6.6. Stream Reset Extension

The SCTP stream reset extension defined in [RFC6525] can be used with DTLS encapsulation. It is used to reset SCTP streams and add SCTP streams during the lifetime of the SCTP association.

6.7. Interleaving of Large User Messages

SCTP as defined in [RFC4960] does not support the interleaving of large user messages that need to be fragmented and reassembled by the SCTP layer. The protocol extension defined in [I-D.ietf-tsvwg-sctp-ndata] overcomes this limitation and can be used with DTLS encapsulation.

7. IANA Considerations

This document requires no actions from IANA.

8. Security Considerations

Security considerations for DTLS are specified in [RFC4347] and for SCTP in [RFC4960], [RFC3758], and [RFC6525]. The combination of SCTP and DTLS introduces no new security considerations.

SCTP should not process the IP addresses used for the underlying communication since DTLS provides no guarantees about them.

It should be noted that the inability to process ICMP or ICMPv6 messages does not add any security issue. When SCTP is carried over a connection-less lower layer like IPv4, IPv6, or UDP, processing of these messages is required to protect other nodes not supporting SCTP. Since DTLS provides a connection-oriented lower layer, this kind of protection is not necessary.

9. Acknowledgments

The authors wish to thank David Black, Benoit Claise, Spencer Dawkins, Francis Dupont, Gorry Fairhurst, Stephen Farrell, Christer Holmberg, Barry Leiba, Eric Rescorla, Tom Taylor, Joe Touch and Magnus Westerlund for their invaluable comments.

10. References

10.1. Normative References

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

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

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, March 2007.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

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

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.

10.2. Informative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, August 2007.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, February 2012.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013.

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-13 (work in progress), November 2014.

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ietf-tsvwg-sctp-prpolicies]
Tuexen, M., Seggelmann, R., Stewart, R., and S. Loreto, "Additional Policies for the Partial Reliability Extension of the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-prpolicies-06 (work in progress), December 2014.

[I-D.ietf-tsvwg-sctp-ndata]
Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "Stream Schedulers and a New Data Chunk for the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-ndata-02 (work in progress), January 2015.

Appendix A. NOTE to the RFC-Editor

Although the references to [I-D.ietf-tsvwg-sctp-prpolicies] and [I-D.ietf-tsvwg-sctp-ndata] are informative, put this document in REF-HOLD until these two references have been approved and update these references to the corresponding RFCs.

Authors' Addresses

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

Email: tuexen@fh-muenster.de

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
US

Email: randall@lakerest.net

Randell Jesup
WorldGate Communications
3800 Horizon Blvd, Suite #103
Trevose, PA 19053-4947
US

Phone: +1-215-354-5166
Email: randell_ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: Salvatore.Loreto@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 20, 2016

Y. Nishida
GE Global Research
P. Natarajan
Cisco Systems
A. Caro
BBN Technologies
P. Amer
University of Delaware
K. Nielsen
Ericsson

February 17, 2016

SCTP-PF: Quick Failover Algorithm in SCTP
draft-ietf-tsvwg-sctp-failover-16.txt

Abstract

SCTP supports multi-homing. However, when the failover operation specified in RFC4960 is followed, there can be significant delay and performance degradation in the data transfer path failover. To overcome this problem this document specifies a quick failover algorithm (SCTP-PF) based on the introduction of a Potentially Failed (PF) state in SCTP Path Management.

The document also specifies a dormant state operation of SCTP. This dormant state operation is required to be followed by an SCTP-PF implementation, but it may equally well be applied by a standard RFC4960 SCTP implementation.

Additionally, the document introduces an alternative switchback operation mode called Primary Path Switchover that will be beneficial in certain situations. This mode of operation applies to both a standard RFC4960 SCTP implementation as well as to a SCTP-PF implementation.

The procedures defined in the document require only minimal modifications to the RFC4960 specification. The procedures are sender-side only and do not impact the SCTP receiver.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Stream Control Transmission Protocol (SCTP) specified in [RFC4960] supports multi-homing at the transport layer. SCTP's multi-homing features include failure detection and failover procedures to provide network interface redundancy and improved end-to-end fault tolerance. In SCTP's current failure detection procedure, the sender must experience Path.Max.Retrans (PMR) number of consecutive failed timer-based retransmissions on a destination address before detecting a path failure. Until detecting the path failure, the sender continues to transmit data on the failed path. The prolonged time in which [RFC4960] SCTP continues to use a failed path severely degrades the performance of the protocol. To address this problem, this document specifies a quick failover algorithm (SCTP-PF) based on the introduction of a new Potentially Failed (PF) path state in SCTP path management. The performance deficiencies of the [RFC4960] failover operation, and the improvements obtainable from the introduction of a Potentially Failed state in SCTP, were proposed and documented in [NATARAJAN09] for Concurrent Multipath Transfer SCTP [IYENGAR06].

While SCTP-PF can accelerate failover process and improve performance, the risks that an SCTP endpoint enters the dormant state where all destination addresses are inactive can be increased. [RFC4960] leaves the protocol operation during dormant state to implementations and encourages to avoid entering the state as much as possible by careful tuning of the Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) parameters. We specify a dormant state operation for SCTP-PF which makes SCTP-PF provide the same disruption tolerance as [RFC4960] despite that the dormant state may be entered more quickly. The dormant state operation may equally well be applied by an [RFC4960] implementation and will here serve to provide added fault tolerance for situations where the tuning of the Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) parameters fail to provide adequate prevention of the entering of the dormant state.

The operation after the recovery of a failed path also impacts the performance of the protocol. With the procedures specified in

[RFC4960] SCTP will, after a failover from the primary path, switch back to use the primary path for data transfer as soon as this path becomes available again. From a performance perspective such a forced switchback of the data transmission path can be suboptimal as the CWND towards the original primary destination address has to be rebuilt once data transfer resumes, [CARO02]. As an optional alternative to the switchback operation of [RFC4960], this document specifies an alternative Primary Path Switchover procedure which avoid such forced switchbacks of the data transfer path. The Primary Path Switchover operation was originally proposed in [CARO02].

While SCTP-PF primarily is motivated by a desire to improve the multi-homed operation, the feature applies also to SCTP single-homed operation. Here the algorithm serves to provide increased failure detection on idle associations, whereas the failover or switchback aspects of the algorithm will not be activated. This is discussed in more detail in Appendix C.

A brief description of the motivation for the introduction of the Potentially Failed state including a discussion of alternative approaches to mitigate the deficiencies of the [RFC4960] failover operation are given in the Appendices. Discussion of path bouncing effects that might be caused by frequent switchovers, are also provided there.

2. Conventions and Terminology

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

3. SCTP with Potentially Failed Destination State (SCTP-PF)

3.1. Overview

To minimize the performance impact during failover, the sender should avoid transmitting data to a failed destination address as early as possible. In the [RFC4960] SCTP path management scheme, the sender stops transmitting data to a destination address only after the destination address is marked inactive. This process takes a significant amount of time as it requires the error counter of the destination address to exceed the Path.Max.Retrans (PMR) threshold. The issue cannot simply be mitigated by lowering of the PMR threshold because this may result in spurious failure detection and unnecessary prevention of the usage of a preferred primary path. Also due to the coupled tuning of the Path.Max.Retrans (PMR) and the Association.Max.Retrans (AMR) parameter values in [RFC4960], lowering

of the PMR threshold may result in lowering of the AMR threshold, which would result in decrease of the fault tolerance of SCTP.

The solution provided in this document is to extend the SCTP path management scheme of [RFC4960] by the addition of the Potentially Failed (PF) state as an intermediate state in between the active and inactive state of a destination address in the [RFC4960] path management scheme, and let the failover of data transfer away from a destination address be driven by the entering of the PF state instead of by the entering of the inactive state. Thereby SCTP may perform quick failover without negatively impacting the overall fault tolerance of [RFC4960] SCTP. At the same time, RTO-based HEARTBEAT probing is initiated towards a destination address once it enters PF state. Thereby SCTP may quickly ascertain whether network connectivity towards the destination address is broken or whether the failover was spurious. In the case where the failover was spurious data transfer may quickly resume towards the original destination address.

The new failure detection algorithm assumes that loss detected by a timeout implies either severe congestion or network connectivity failure. It recommends that by default a destination address is classified as PF at the occurrence of the first timeout.

3.2. Specification of the SCTP-PF Procedures

The SCTP-PF operation is specified as follows:

1. The sender maintains a new tunable SCTP Protocol Parameter called PotentiallyFailed.Max.Retrans (PFMR). The PFMR defines the new intermediate PF threshold on the destination address error counter. When this threshold is exceeded the destination address is classified as PF. The RECOMMENDED value of PFMR is 0. If PFMR is set to be greater than or equal to Path.Max.Retrans (PMR), the resulting PF threshold will be so high that the destination address will reach the inactive state before it can be classified as PF.

2. The error counter of an active destination address is incremented or cleared as specified in [RFC4960]. This means that the error counter of the destination address in active state will be incremented each time the T3-rtx timer expires, or each time a HEARTBEAT chunk is sent when idle and not acknowledged within an RTO. When the value in the destination address error counter exceeds PFMR, the endpoint MUST mark the destination address as in the PF state.

3. A SCTP-PF sender SHOULD NOT send data to destination addresses in PF state when alternative destination addresses in active state are available. Specifically this means that:

   i When there is outbound data to send and the destination address presently used for data transmission is in PF state, the sender SHOULD choose a destination address in active state, if one exists, and use this destination address for data transmission.

   ii As specified in [RFC4960] section 6.4.1, when the sender retransmits data that has timed out, it should attempt to pick a new destination address for data retransmission. In this case, the sender SHOULD choose an alternate destination transport address in active state if one exists.

   iii When there is outbound data to send and the SCTP user explicitly requests to send data to a destination address in PF state, the sender SHOULD send the data to an alternate destination address in active state if one exists.

   When choosing among multiple destination addresses in active state an SCTP sender will follow the guiding principles of section 6.4.1 of [RFC4960] of choosing most divergent source-destination pairs compared with, for i.: the destination address in PF state that it performs a failover from, and for ii.: the destination address towards which the data timed out. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document.

   In all cases, the sender MUST NOT change the state of chosen destination address, whether this state be active or PF, and it MUST NOT clear the error counter of the destination address as a result of choosing the destination address for data transmission.

4. When the destination addresses are all in PF state or some in PF state and some in inactive state, the sender MUST choose one destination address in PF state and SHOULD transmit or retransmit data to this destination address using the following rules:

   A. The sender SHOULD choose the destination in PF state with the lowest error count (fewest consecutive timeouts) for data transmission and transmit or retransmit data to this destination.

B. When there are multiple destination addresses in PF state with same error count, the sender should let the choice among the multiple destination addresses in PF state with equal error count be based on the [RFC4960], section 6.4.1, principles of choosing most divergent source-destination pairs when executing (potentially consecutive) retransmission. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document.

The sender MUST NOT change the state and the error counter of any destination addresses as the result of the selection.

5. The HB.interval of the Path Heartbeat function of [RFC4960] MUST be ignored for destination addresses in PF state. Instead HEARTBEAT chunks are sent to destination addresses in PF state once per RTO. HEARTBEAT chunks SHOULD be sent to destination addresses in PF state, but the sending of HEARTBEATS MUST honor whether the Path Heartbeat function (Section 8.3 of [RFC4960]) is enabled for the destination address or not. I.e., if the Path Heartbeat function is disabled for the destination address in question, HEARTBEATS MUST NOT be sent. Note that when Heartbeat function is disabled, it may take longer to transition a destination address in PF state back to active state.

6. HEARTBEATs are sent when a destination address reaches the PF state. When a HEARTBEAT chunk is not acknowledged within the RTO, the sender increments the error counter and exponentially backs off the RTO value. If the error counter is less than PMR, the sender transmits another packet containing the HEARTBEAT chunk immediately after timeout expiration on the previous HEARTBEAT. When data is being transmitted to a destination address in the PF state, the transmission of a HEARTBEAT chunk MAY be omitted in case where the receipt of a SACK of the data or a T3-rtx timer expiration on the data can provide equivalent information, such as the case where the data chunk has been transmitted to a single destination address only. Likewise, the timeout of a HEARTBEAT chunk MAY be ignored if data is outstanding towards the destination address.

7. When the sender receives a HEARTBEAT ACK from a HEARTBEAT sent to a destination address in PF state, the sender SHOULD clear the error counter of the destination address and transition the destination address back to active state. However, there may be a situation where HEARTBEAT chunks can go through while DATA chunks cannot. Hence, in a situation where a HEARTBEAT ACK arrives while there is data outstanding towards the destination address to which the HEARTBEAT was sent, then an implementation

MAY choose to not have the HEARTBEAT ACK reset the error counter, but have the error counter reset await the fate of the outstanding data transmission. This situation can happen when data is sent to a destination address in PF state. When the sender resumes data transmission on a destination address after a transition of the destination address from PF to active state, it MUST do this following the prescriptions of Section 7.2 of [RFC4960].

8. Additional (PMR - PFMR) consecutive timeouts on a destination address in PF state confirm the path failure, upon which the destination address transitions to the inactive state. As described in [RFC4960], the sender (i) SHOULD notify the ULP about this state transition, and (ii) transmit HEARTBEAT chunks to the inactive destination address at a lower HB.interval frequency as described in Section 8.3 of [RFC4960] (when the Path Heartbeat function is enabled for the destination address).

9. Acknowledgments for chunks that have been transmitted to multiple destinations (i.e., a chunk which has been retransmitted to a different destination address than the destination address to which the chunk was first transmitted) SHOULD NOT clear the error count for an inactive destination address and SHOULD NOT move a destination address in PF state back to active state, since a sender cannot disambiguate whether the ACK was for the original transmission or the retransmission(s). A SCTP sender MAY clear the error counter and move a destination address back to active state by information other than acknowledgments, when it can uniquely determine which destination, among multiple destination addresses, the chunk reached. This document makes no reference to what such information could consist of, nor how such information could be obtained.

10. Acknowledgments for data chunks that has been transmitted to one destination address only MUST clear the error counter for the destination address and MUST transition a destination address in PF state back to active state. This situation can happen when new data is sent to a destination address in the PF state. It can also happen in situations where the destination address is in the PF state due to the occurrence of a spurious T3-rtx timer and acknowledgments start to arrive for data sent prior to occurrence of the spurious T3-rtx and data has not yet been retransmitted towards other destinations. This document does not specify special handling for detection of or reaction to spurious T3-rtx timeouts, e.g., for special operation vis-a-vis the congestion control handling or data retransmission operation towards a destination address which undergoes a transition from

active to PF to active state due to a spurious T3-rtx timeout. But it is noted that this is an area which would benefit from additional attention, experimentation and specification for single-homed SCTP as well as for multi-homed SCTP protocol operation.

11. When all destination addresses are in inactive state, and SCTP protocol operation thus is said to be in dormant state, the prescriptions given in Section 4 shall be followed.

12. The SCTP stack SHOULD expose the PF state of its destination addresses to the ULP as well as provide the means to notify the ULP of state transitions of its destination addresses from active to PF, and vice-versa. However it is recommended that an SCTP stack implementing SCTP-PF also allows for that the ULP is kept ignorant of the PF state of its destinations and the associated state transitions, thus allowing for retain of the simpler state transition model of RFC4960 in the ULP. For this reason it is recommended that an SCTP stack implementing SCTP-PF also provides the ULP with the means to suppress exposure of the PF state and the associated state transitions.

4. Dormant State Operation

In a situation with complete disruption of the communication in between the SCTP Endpoints, the aggressive HEARTBEAT transmissions of SCTP-PF on destination addresses in PF state may make the association enter dormant state faster than a standard [RFC4960] SCTP implementation given the same setting of Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR). For example, an SCTP association with two destination addresses typically would reach dormant state in half the time of an [RFC4960] SCTP implementation in such situations. This is because a SCTP PF sender will send HEARTBEATS and data retransmissions in parallel with RTO intervals when there are multiple destinations addresses in PF state. This argument presumes that RTO << HB.interval of [RFC4960]. With the design goal that SCTP-PF shall provide the same level of disruption tolerance as an [RFC4960] SCTP implementation with the same Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) setting, we prescribe for that an SCTP-PF implementation SHOULD operate as described below in Section 4.1 during dormant state.

An SCTP-PF implementation MAY choose a different dormant state operation than the one described below in Section 4.1 provided that the solution chosen does not decrease the fault tolerance of the SCTP-PF operation.

The below prescription for SCTP-PF dormant state handling MUST NOT be coupled to the value of the PFMR, but solely to the activation of SCTP-PF logic in an SCTP implementation.

It is noted that the below dormant state operation is considered to provide added disruption tolerance also for an [RFC4960] SCTP implementation, and that it can be sensible for an [RFC4960] SCTP implementation to follow this mode of operation. For an [RFC4960] SCTP implementation the continuation of data transmission during dormant state makes the fault tolerance of SCTP be more robust towards situations where some, or all, alternative paths of an SCTP association approach, or reach, inactive state before the primary path used for data transmission observes trouble.

4.1. SCTP Dormant State Procedure

a. When the destination addresses are all in inactive state and data is available for transfer, the sender MUST choose one destination and transmit data to this destination address.

b. The sender MUST NOT change the state of the chosen destination address (it remains in inactive state) and it MUST NOT clear the error counter of the destination address as a result of choosing the destination address for data transmission.

c. The sender SHOULD choose the destination in inactive state with the lowest error count (fewest consecutive timeouts) for data transmission. When there are multiple destinations with same error count in inactive state, the sender SHOULD attempt to pick the most divergent source - destination pair from the last source - destination pair where failure was observed. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document. To support differentiation of inactive destination addresses based on their error count SCTP will need to allow for increment of the destination address error counters up to some reasonable limit above PMR+1, thus changing the prescriptions of [RFC4960], section 8.3, in this respect. The exact limit to apply is not specified in this document but it is considered reasonable to require for the limit to be an order of magnitude higher than the PMR value. A sender MAY choose to deploy other strategies that the strategy defined here. The strategy to prioritize the last active destination address, i.e., the destination address with the fewest error counts is optimal when some paths are permanently inactive, but suboptimal when a path instability is transient.

5. Primary Path Switchover

The objective of the Primary Path Switchover operation is to allow the SCTP sender to continue data transmission on a new working path even when the old primary destination address becomes active again. This is achieved by having SCTP perform a switchover of the primary path to the new working path if the error counter of the primary path exceeds a certain threshold. This mode of operation can be applied not only to SCTP-PF implementations, but also to [RFC4960] implementations.

The Primary Path Switchover operation requires only sender side changes. The details are:

1. The sender maintains a new tunable parameter, called Primary.Switchover.Max.Retrans (PSMR). For SCTP-PF implementations, the PSMR MUST be set greater or equal to the PFMR value. For [RFC4960] implementations the PSMR MUST be set greater or equal to the PMR value. Implementations MUST reject any other values of PSMR.

2. When the path error counter on a set primary path exceeds PSMR, the SCTP implementation MUST autonomously select and set a new primary path.

3. The primary path selected by the SCTP implementation MUST be the path which at the given time would be chosen for data transfer. A previously failed primary path can be used as data transfer path as per normal path selection when the present data transfer path fails.

4. For SCTP-PF, the recommended value of PSMR is PFMR when Primary Path Switchover operation mode is used. This means that no forced switchback to a previously failed primary path is performed. An SCTP-PF implementation of Primary Path Switchover MUST support the setting of PSMR = PFMR. A SCTP-PF implementation of Primary Path Switchover MAY support setting of PSMR > PFMR.

5. For [RFC4960] SCTP, the recommended value of PSMR is PMR when Primary Path Switchover is used. This means that no forced switchback to a previously failed primary path is performed. A [RFC4960] SCTP implementation of Primary Path Switchover MUST support the setting of PSMR = PMR. An [RFC4960] SCTP implementation of Primary Path Switchover MAY support larger settings of PSMR > PMR.

6. It MUST be possible to disable the Primary Path Switchover operation and obtain the standard switchback operation of [RFC4960].

The manner of switchover operation that is most optimal in a given scenario depends on the relative quality of a set primary path versus the quality of alternative paths available as well as on the extent to which it is desired for the mode of operation to enforce traffic distribution over a number of network paths. I.e., load distribution of traffic from multiple SCTP associations may be sought to be enforced by distribution of the set primary paths with [RFC4960] switchback operation. However as [RFC4960] switchback behavior is suboptimal in certain situations, especially in scenarios where a number of equally good paths are available, an SCTP implementation MAY support also, as alternative behavior, the Primary Path Switchover mode of operation and MAY enable it based on applications' requests.

For an SCTP implementation that implements the Primary Path Switchover operation, this specification RECOMMENDS that the standard RFC4960 switchback operation is retained as the default operation.

6. Suggested SCTP Protocol Parameter Values

This document does not alter the [RFC4960] value recommendation for the SCTP Protocol Parameters defined in [RFC4960].

The following protocol parameter is RECOMMENDED:

PotentiallyFailed.Max.Retrans (PFMR) - 0

7. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to control and observe the SCTP-PF behavior as well as the Primary Path Switchover function.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is, by means of the existing SCTP_PEER_ADDR_CHANGE event, extended to provide the event notification when a peer address enters or leaves the potentially failed state as well as the socket API implementation is extended to expose the potentially failed state of a peer address in the existing SCTP_GET_PEER_ADDR_INFO structure.

Furthermore, two new read/write socket options for the level IPPROTO_SCTP and the name SCTP_PEER_ADDR_THLDS and

SCTP_EXPOSE_POTENTIALLY_FAILED_STATE are defined as described below. The first socket option is used to control the values of the PFMR and PSMR parameters described in Section 3 and in Section 5. The second one controls the exposition of the potentially failed path state.

Support for the SCTP_PEER_ADDR_THLDS and SCTP_EXPOSE_POTENTIALLY_FAILED_STATE socket options need also to be added to the function sctp_opt_info().

7.1. Support for the Potentially Failed Path State

As defined in [RFC6458], the SCTP_PEER_ADDR_CHANGE event is provided if the status of a peer address changes. In addition to the state changes described in [RFC6458], this event is also provided, if a peer address enters or leaves the potentially failed state. The notification as defined in [RFC6458] uses the following structure:

```
struct sctp_paddr_change {
  uint16_t spc_type;
  uint16_t spc_flags;
  uint32_t spc_length;
  struct sockaddr_storage spc_aaddr;
  uint32_t spc_state;
  uint32_t spc_error;
  sctp_assoc_t spc_assoc_id;
}
```

[RFC6458] defines the constants SCTP_ADDR_AVAILABLE, SCTP_ADDR_UNREACHABLE, SCTP_ADDR_REMOVED, SCTP_ADDR_ADDED, and SCTP_ADDR_MADE_PRIM to be provided in the spc_state field. This document defines in addition to that the new constant SCTP_ADDR_POTENTIALLY_FAILED, which is reported if the affected address becomes potentially failed.

The SCTP_GET_PEER_ADDR_INFO socket option defined in [RFC6458] can be used to query the state of a peer address. It uses the following structure:

```
struct sctp_paddrinfo {
  sctp_assoc_t spinfo_assoc_id;
  struct sockaddr_storage spinfo_address;
  int32_t spinfo_state;
  uint32_t spinfo_cwnd;
  uint32_t spinfo_srtt;
  uint32_t spinfo_rto;
  uint32_t spinfo_mtu;
};
```

[RFC6458] defines the constants SCTP_UNCONFIRMED, SCTP_ACTIVE, and SCTP_INACTIVE to be provided in the spinfo_state field. This document defines in addition to that the new constant SCTP_POTENTIALLY_FAILED, which is reported if the peer address is potentially failed.

7.2. Peer Address Thresholds (SCTP_PEER_ADDR_THLDS) Socket Option

Applications can control the SCTP-PF behavior by getting or setting the number of consecutive timeouts before a peer address is considered potentially failed or unreachable. The same socket option is used by applications to set and get the number of timeouts before the primary path is changed automatically by the Primary Path Switchover function. This socket option uses the level IPPROTO_SCTP and the name SCTP_PEER_ADDR_THLDS.

The following structure is used to access and modify the thresholds:

```
struct sctp_paddrthlds {
  sctp_assoc_t spt_assoc_id;
  struct sockaddr_storage spt_address;
  uint16_t spt_pathmaxrxt;
  uint16_t spt_pathpfthld;
  uint16_t spt_pathcpthld;
};
```

spt_assoc_id: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application may fill in an association identifier or SCTP_FUTURE_ASSOC. It is an error to use SCTP_{CURRENT|ALL}_ASSOC in spt_assoc_id.

spt_address: This specifies which peer address is of interest. If a wild card address is provided, this socket option applies to all current and future peer addresses.

spt_pathmaxrxt: Each peer address of interest is considered unreachable, if its path error counter exceeds spt_pathmaxrxt.

spt_pathpfthld: Each peer address of interest is considered Potentially Failed, if its path error counter exceeds spt_pathpfthld.

spt_pathcpthld: Each peer address of interest is not considered the primary remote address anymore, if its path error counter exceeds spt_pathcpthld. Using a value of 0xffff disables the selection of a new primary peer address. If an implementation does not support the automatically selection of a new primary address, it should indicate an error with errno set to EINVAL if a value different

from 0xffff is used in spt_pathcpthld. For SCTP-PF, the setting of spt_pathcpthld < spt_pathpfthld should be rejected with errno set to EINVAL. For [RFC4960] SCTP, the setting of spt_pathcpthld < spt_pathmaxrxt should be rejected with errno set to EINVAL. A SCTP-PF implementation may support only setting of spt_pathcpthld = spt_pathpfthld and spt_pathcpthld = 0xffff and a [RFC4960] SCTP implementation may support only setting of spt_pathcpthld = spt_pathmaxrxt and spt_pathcpthld = 0xffff. In these cases SCTP shall reject setting of other values with errno set to EINVAL.

7.3. Exposing the Potentially Failed Path State (SCTP_EXPOSE_POTENTIALLY_FAILED_STATE) Socket Option

Applications can control the exposure of the potentially failed path state in the SCTP_PEER_ADDR_CHANGE event and the SCTP_GET_PEER_ADDR_INFO as described in Section 7.1. The default value is implementation specific.

This socket option uses the level IPPROTO_SCTP and the name SCTP_EXPOSE_POTENTIALLY_FAILED_STATE.

The following structure is used to control the exposition of the potentially failed path state:

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application may fill in an association identifier or SCTP_FUTURE_ASSOC. It is an error to use SCTP_{CURRENT|ALL}_ASSOC in assoc_id.

assoc_value: The potentially failed path state is exposed if and only if this parameter is non-zero.

8. Security Considerations

Security considerations for the use of SCTP and its APIs are discussed in [RFC4960] and [RFC6458].

The logic introduced by this document does not impact existing SCTP messages on the wire. Also, this document does not introduce any new SCTP messages on the wire that require new security considerations.

SCTP-PF makes SCTP not only more robust during primary path failure/ congestion but also more vulnerable to network connectivity/

congestion attacks on the primary path. SCTP-PF makes it easier for an attacker to trick SCTP to change data transfer path, since the duration of time that an attacker needs to negatively influence the network connectivity is much shorter than [RFC4960]. However, SCTP-PF does not constitute a significant change in the duration of time and effort an attacker needs to keep SCTP away from the primary path. With the standard switchback operation [RFC4960] SCTP resumes data transfer on its primary path as soon as the next HEARTBEAT succeeds.

On the other hand, usage of the Primary Path Switchover mechanism, does change the threat analysis. This is because on-path attackers can force a permanent change of the data transfer path by blocking the primary path until the switchover of the primary path is triggered by the Primary Path Switchover algorithm. This especially will be the case when the Primary Path Switchover is used together with SCTP-PF with the particular setting of PSMR = PFMR = 0, as Primary Path Switchover here happens already at the first RTO timeout experienced. Users of the Primary Path Switchover mechanism should be aware of this fact.

The event notification of path state transfer from active to potentially failed state and vice versa gives attackers an increased possibility to generate more local events. However, it is assumed that event notifications are rate-limited in the implementation to address this threat.

9. MIB Considerations

SCTP-PF introduces new SCTP algorithms for failover and switchback with associated new state parameters. It is recommended that the SCTP-MIB defined in [RFC3873] is updated to support the management of the SCTP-PF implementation. This can be done by extending the sctpAssocRemAddrActive field of the SCTPAssocRemAddrTable to include information of the PF state of the destination address and by adding new fields to the SCTPAssocRemAddrTable supporting PotentiallyFailed.Max.Retrans (PFMR) and Primary.Switchover.Max.Retrans (PSMR) parameters.

10. IANA Considerations

This document does not create any new registries or modify the rules for any existing registries managed by IANA.

11. Acknowledgements

The authors wish to thank Michael Tuexen for his many invaluable comments and for his very substantial support with the making of this document.

12. Proposed Change of Status (to be Deleted before Publication)

Initially this work looked to entail some changes of the Congestion Control (CC) operation of SCTP and for this reason the work was proposed as Experimental. These intended changes of the CC operation have since been judged to be irrelevant and are no longer part of the specification. As the specification entails no other potential harmful features, consensus exists in the WG to bring the work forward as PS.

Initially concerns have been expressed about the possibility for the mechanism to introduce path bouncing with potential harmful network impacts. These concerns are believed to be unfounded. This issue is addressed in Appendix B.

It is noted that the feature specified by this document is implemented by multiple SCTP SW implementations and furthermore that various variants of the solution have been deployed in telephony signaling environments for several years with good results.

13. References

13.1. Normative References

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

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

13.2. Informative References

[CARO02] Caro Jr., A., Iyengar, J., Amer, P., Heinz, G., and R. Stewart, "A Two-level Threshold Recovery Mechanism for SCTP", Tech report, CIS Dept, University of Delaware , 7 2002.

[CARO04] Caro Jr., A., Amer, P., and R. Stewart, "End-to-End Failover Thresholds for Transport Layer Multi homing", MILCOM 2004 , 11 2004.

[CARO05] Caro Jr., A., "End-to-End Fault Tolerance using Transport Layer Multi homing", Ph.D Thesis, University of Delaware , 1 2005.

[FALLON08]
Fallon, S., Jacob, P., Qiao, Y., Murphy, L., Fallon, E., and A. Hanley, "SCTP Switchover Performance Issues in WLAN Environments", IEEE CCNC 2008, 1 2008.

[GRINNEMO04]
Grinnemo, K-J. and A. Brunstrom, "Performance of SCTP-controlled failovers in M3UA-based SIGTRAN networks", Advanced Simulation Technologies Conference , 4 2004.

[IYENGAR06]
Iyengar, J., Amer, P., and R. Stewart, "Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-end Paths.", IEEE/ACM Trans on Networking 14(5), 10 2006.

[JUNGMAIER02]
Jungmaier, A., Rathgeb, E., and M. Tuexen, "On the use of SCTP in failover scenarios", World Multiconference on Systemics, Cybernetics and Informatics , 7 2002.

[NATARAJAN09]
Natarajan, P., Ekiz, N., Amer, P., and R. Stewart, "Concurrent Multipath Transfer during Path Failure", Computer Communications , 5 2009.

[RFC3873] Pastor, J. and M. Belinchon, "Stream Control Transmission Protocol (SCTP) Management Information Base (MIB)", RFC 3873, DOI 10.17487/RFC3873, September 2004, <http://www.rfc-editor.org/info/rfc3873>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, December 2011.

Appendix A. Discussions of Alternative Approaches

This section lists alternative approaches for the issues described in this document. Although these approaches do not require to update RFC4960, we do not recommend them from the reasons described below.

A.1. Reduce Path.Max.Retrans (PMR)

Smaller values for Path.Max.Retrans shorten the failover duration and in fact this is recommended in some research results [JUNGMAIER02] [GRINNEMO04] [FALLON08]. However to significantly reduce the failover time it is required to go down (as with PFMR) to Path.Max.Retrans=0 and with this setting SCTP switches to another

destination address already on a single timeout which may result in spurious failover. Spurious failover is a problem in [RFC4960] SCTP as the transmission of HEARTBEATS on the left primary path, unlike in SCTP-PF, is governed by 'HB.interval' also during the failover process. 'HB.interval' is usually set in the order of seconds (recommended value is 30 seconds) and when the primary path becomes inactive, the next HEARTBEAT may be transmitted only many seconds later. Indeed as recommended, only 30 secs later. Meanwhile, the primary path may since long have recovered, if it needed recovery at all (indeed the failover could be truly spurious). In such situations, post failover, an endpoint is forced to wait in the order of many seconds before the endpoint can resume transmission on the primary path and furthermore once it returns on the primary path the CWND needs to be rebuild anew - a process which the throughput already have had to suffer from on the alternate path. Using a smaller value for 'HB.interval' might help this situation, but it would result in a general waste of bandwidth as such more frequent HEARTBEATING would take place also when there are no observed troubles. The bandwidth overhead may be diminished by having the ULP use a smaller 'HB.interval' only on the path which at any given time is set to be the primary path, but this adds complication in the ULP.

In addition, smaller Path.Max.Retrans values also affect the 'Association.Max.Retrans' value. When the SCTP association's error count exceeds Association.Max.Retrans threshold, the SCTP sender considers the peer endpoint unreachable and terminates the association. Section 8.2 in [RFC4960] recommends that Association.Max.Retrans value should not be larger than the summation of the Path.Max.Retrans of each of the destination addresses. Else the SCTP sender considers its peer reachable even when all destinations are INACTIVE and to avoid this dormant state operation, [RFC4960] SCTP implementation SHOULD reduce Association.Max.Retrans accordingly whenever it reduces Path.Max.Retrans. However, smaller Association.Max.Retrans value decreases the fault tolerance of SCTP as it increases the chances of association termination during minor congestion events.

A.2. Adjust RTO related parameters

As several research results indicate, we can also shorten the duration of failover process by adjusting RTO related parameters [JUNGMAIER02] [FALLON08]. During failover process, RTO keeps being doubled. However, if we can choose smaller value for RTO.max, we can stop the exponential growth of RTO at some point. Also, choosing smaller values for RTO.initial or RTO.min can contribute to keep the RTO value small.

Similar to reducing Path.Max.Retrans, the advantage of this approach is that it requires no modification to the current specification, although it needs to ignore several recommendations described in the Section 15 of [RFC4960]. However, this approach requires to have enough knowledge about the network characteristics between end points. Otherwise, it can introduce adverse side-effects such as spurious timeouts.

The significant issue with this approach, however, is that even if the RTO.max is lowered to an optimal low value, then as long as the Path.Max.Retrans is kept at the [RFC4960] recommended value, the reduction of the RTO.max doesn’t reduce the failover time sufficiently enough to prevent severe performance degradation during failover.

Appendix B. Discussions for Path Bouncing Effect

The methods described in the document can accelerate the failover process. Hence, they might introduce the path bouncing effect where the sender keeps changing the data transmission path frequently. This sounds harmful to the data transfer, however several research results indicate that there is no serious problem with SCTP in terms of path bouncing effect [CARO04] [CARO05].

There are two main reasons for this. First, SCTP is basically designed for multipath communication, which means SCTP maintains all path related parameters (CWND, ssthresh, RTT, error count, etc) per each destination address. These parameters cannot be affected by path bouncing. In addition, when SCTP migrates the data transfer to another path, it starts with the minimal or the initial CWND. Hence, there is little chance for packet reordering or duplicating.

Second, even if all communication paths between the end-nodes share the same bottleneck, the SCTP-PF results in a behavior already allowed by [RFC4960].

Appendix C. SCTP-PF for SCTP Single-homed Operation

For a single-homed SCTP association the only tangible effect of the activation of SCTP-PF operation is enhanced failure detection in terms of potential notification of the PF state of the sole destination address as well as, for idle associations, more rapid entering, and notification, of inactive state of the destination address and more rapid end-point failure detection. It is believed that neither of these effects are harmful, provided adequate dormant state operation is implemented, and furthermore that they may be particularly useful for applications that deploys multiple SCTP associations for load balancing purposes. The early notification of

the PF state may be used for preventive measures as the entering of the PF state can be used as a warning of potential congestion. Depending on the PMR value, the aggressive HEARTBEAT transmission in PF state may speed up the end-point failure detection (exceed of AMR threshold on the sole path error counter) on idle associations in case where relatively large HB.interval value compared to RTO (e.g. 30secs) is used.

Authors' Addresses

Yoshifumi Nishida
GE Global Research
2623 Camino Ramon
San Ramon, CA 94583
USA

Email: nishida@wide.ad.jp

Preethi Natarajan
Cisco Systems
510 McCarthy Blvd
Milpitas, CA 95035
USA

Email: prenatar@cisco.com

Armando Caro
BBN Technologies
10 Moulton St.
Cambridge, MA 02138
USA

Email: acaro@bbn.com

Paul D. Amer
University of Delaware
Computer Science Department - 434 Smith Hall
Newark, DE 19716-2586
USA

Email: amer@udel.edu

Karen E. E. Nielsen
Ericsson
Kistavaegen 25
Stockholm 164 80
Sweden

Email: karen.nielsen@tieto.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 5, 2018

R. Stewart
Netflix, Inc.
M. Tuexen
Muenster Univ. of Appl. Sciences
S. Loreto
Ericsson
R. Seggelmann
Metafinanz Informationssysteme GmbH

September 1, 2017

Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol

draft-ietf-tsvwg-sctp-ndata-13.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a message oriented transport protocol supporting arbitrarily large user messages. This document adds a new chunk to SCTP for carrying payload data. This allows a sender to interleave different user messages that would otherwise result in head of line blocking at the sender. The interleaving of user messages is required for WebRTC Datachannels.

Whenever an SCTP sender is allowed to send user data, it may choose from multiple outgoing SCTP streams. Multiple ways for performing this selection, called stream schedulers, are defined in this document. A stream scheduler can choose to either implement, or not implement, user message interleaving.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Overview

When SCTP [RFC4960] was initially designed it was mainly envisioned for the transport of small signaling messages. Late in the design stage it was decided to add support for fragmentation and reassembly of larger messages with the thought that someday Session Initiation Protocol (SIP) [RFC3261] style signaling messages may also need to use SCTP and a single Maximum Transmission Unit (MTU) sized message would be too small. Unfortunately this design decision, though valid at the time, did not account for other applications that might send large messages over SCTP. The sending of such large messages over SCTP as specified in [RFC4960] can result in a form of sender side head of line blocking (e.g., when the transmission of a message is blocked from transmission because the sender has started the transmission of another, possibly large, message). This head of line blocking is caused by the use of the Transmission Sequence Number (TSN) for three different purposes:

1. As an identifier for DATA chunks to provide a reliable transfer.

2. As an identifier for the sequence of fragments to allow reassembly.

3. As a sequence number allowing up to 2**16 - 1 Stream Sequence Numbers (SSNs) outstanding.

The protocol requires all fragments of a user message to have consecutive TSNs. This document allows an SCTP sender to interleave different user messages.

This document also defines several stream schedulers for general SCTP associations allowing different relative stream treatments. The stream schedulers may behave differently depending on whether user message interleaving has been negotiated for the association or not.

Figure 1 illustrates the behaviour of a round robin stream scheduler using DATA chunks when three streams with the Stream Identifiers

(SIDs) 0, 1, and 2 are used. Each queue for SID 0 and SID 2 contains a single user message requiring three chunks, the queue for SID 1 contains three user messages each requiring a single chunk. It is shown how these user messages are encapsulated in chunk using TSN 0 to TSN 8. Please note that the use of such a scheduler implies late TSN assignment but it can be used with an [RFC4960] compliant implementation that does not support user message interleaving. Late TSN assignment means that the sender generates chunks from user messages and assigns the TSN as late as possible in the process of sending the user messages.

```
+---+---+---+
|    0/0    |-+
+---+---+---+ |
              |  +---+---+---+---+---+---+---+---+---+
+---+---+---+ +->|1/2|1/1|2/0|2/0|2/0|1/0|0/0|0/0|0/0|
|1/2|1/1|1/0|--->|---|---|---|---|---|---|---|---|---|
+---+---+---+ +->| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
              |  +---+---+---+---+---+---+---+---+---+
+---+---+---+ |
|    2/0    |-+
+---+---+---+
                                +-------+
  +-------+                     |SID/SSN|
  |SID/SSN|                     |-------|
  +-------+                     |  TSN  |
                                +-------+
```

Figure 1: Round Robin Scheduler without User Message Interleaving

This document describes a new chunk carrying payload data called I-DATA. This chunk incorporates the properties of the current SCTP DATA chunk, all the flags and fields except the Stream Sequence Number (SSN), but also adds two new fields in its chunk header, the Fragment Sequence Number (FSN) and the Message Identifier (MID). The FSN is only used for reassembling all fragments having the same MID and ordering property. The TSN is only used for the reliable transfer in combination with Selective Acknowledgment (SACK) chunks.

In addition, the MID is also used for ensuring ordered delivery instead of using the stream sequence number (The I-DATA chunk omits a SSN.).

Figure 2 illustrates the behaviour of an interleaving round robin stream scheduler using I-DATA chunks.

```
+---+---+---+
|    0/0    |-+
+---+---+---+ |
              |  +-----+-----+-----+-----+-----+-----+-----+-----+-----+
+---+---+---+ +->|2/0/2|1/2/0|0/0/2|2/0/1|1/1/0|0/0/1|2/0/0|1/0/0|0/0/0|
|1/2|1/1|1/0|--->|-----|-----|-----|-----|-----|-----|-----|-----|-----|
+---+---+---+ +->|  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
              |  +-----+-----+-----+-----+-----+-----+-----+-----+-----+
+---+---+---+ |
|    2/0    |-+
+---+---+---+
                                      +-----------+
  +-------+                           |SID/MID/FSN|
  |SID/MID|                           |-----------|
  +-------+                           |    TSN    |
                                      +-----------+
```

Figure 2: Round Robin Scheduler with User Message Interleaving

The support of the I-DATA chunk is negotiated during the association setup using the Supported Extensions Parameter as defined in [RFC5061]. If I-DATA support has been negotiated for an association, I-DATA chunks are used for all user-messages. DATA chunks are not permitted when I-DATA support has been negotiated. It should be noted that an SCTP implementation supporting I-DATA chunks needs to allow the coexistence of associations using DATA chunks and associations using I-DATA chunks.

In Section 2 this document specifies the user message interleaving by defining the I-DATA chunk, the procedures to use it and its interactions with other SCTP extensions. Multiple stream schedulers are defined in Section 3 followed in Section 4 by describing an extension to the socket API for using what is specified in this document.

1.2. Conventions

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

2. User Message Interleaving

The protocol mechanisms described in this document allow the interleaving of user messages sent on different streams. They do not support the interleaving of multiple messages (ordered or unordered) sent on the same stream.

The interleaving of user messages is required for WebRTC Datachannels as specified in [I-D.ietf-rtcweb-data-channel].

An SCTP implementation supporting user message interleaving is REQUIRED to support the coexistence of associations using DATA chunks and associations using I-DATA chunks. If an SCTP implementation supports user message interleaving and the Partial Reliability extension described in [RFC3758] or the Stream Reconfiguration Extension described in [RFC6525], it is REQUIRED to implement the corresponding changes specified in Section 2.3.

2.1. The I-DATA Chunk Supporting User Message Interleaving

The following Figure 3 shows the new I-DATA chunk allowing user message interleaving.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 64   |  Res  |I|U|B|E|       Length = Variable       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              TSN                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Stream Identifier      |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Message Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Payload Protocol Identifier / Fragment Sequence Number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                           User Data                           /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: I-DATA chunk format

The only differences between the I-DATA chunk in Figure 3 and the DATA chunk defined in [RFC4960] and [RFC7053] are the addition of the new Message Identifier (MID) and the new Fragment Sequence Number (FSN) and the removal of the Stream Sequence Number (SSN). The Payload Protocol Identifier (PPID) already defined for DATA chunks in [RFC4960] and the new FSN are stored at the same location of the packet using the B bit to determine which value is stored at the location. The length of the I-DATA chunk header is 20 bytes, which is 4 bytes more than the length of the DATA chunk header defined in [RFC4960] and [RFC7053].

The old fields are:

Res: 4 bits
These bits are reserved. They MUST be set to 0 by the sender and MUST be ignored by the receiver.

I bit: 1 bit
The (I)mmediate Bit, if set, indicates that the receiver SHOULD NOT delay the sending of the corresponding SACK chunk. Same as the I bit for DATA chunks as specified in [RFC7053].

U bit: 1 bit
The (U)nordered bit, if set, indicates the user message is unordered. Same as the U bit for DATA chunks as specified in [RFC4960].

B bit: 1 bit
The (B)eginning fragment bit, if set, indicates the first fragment of a user message. Same as the B bit for DATA chunks as specified in [RFC4960].

E bit: 1 bit
The (E)nding fragment bit, if set, indicates the last fragment of a user message. Same as the E bit for DATA chunks as specified in [RFC4960].

Length: 16 bits (unsigned integer)
This field indicates the length of the DATA chunk in bytes from the beginning of the type field to the end of the User Data field excluding any padding. Similar to the Length for DATA chunks as specified in [RFC4960].

TSN: 32 bits (unsigned integer)
This value represents the TSN for this I-DATA chunk. Same as the TSN for DATA chunks as specified in [RFC4960].

Stream Identifier: 16 bits (unsigned integer)
Identifies the stream to which the user data belongs. Same as the Stream Identifier for DATA chunks as specified in [RFC4960].

The new fields are:

Reserved: 16 bits (unsigned integer)
This field is reserved. It MUST be set to 0 by the sender and MUST be ignored by the receiver.

Message Identifier (MID): 32 bits (unsigned integer)
The MID is the same for all fragments of a user message, it is used to determine which fragments (enumerated by the FSN) belong to the same user message. For ordered user messages, the MID is

also used by the SCTP receiver to deliver the user messages in the correct order to the upper layer (similar to the SSN of the DATA chunk defined in [RFC4960]). The sender uses for each outgoing stream two counters, one for ordered messages, one for unordered messages. All of these counters are independent and initially 0. They are incremented by 1 for each user message. Please note that the serial number arithmetic defined in [RFC1982] using SERIAL_BITS = 32 applies. Therefore, the sender MUST NOT have more than 2**31 - 1 ordered messages for each outgoing stream in flight and MUST NOT have more than 2**31 - 1 unordered messages for each outgoing stream in flight. A message is considered in flight, if at least on of its I-DATA chunks is not acknowledged in a non-renegable way (i.e. not acknowledged by the cumulative TSN Ack). Please note that the MID is in "network byte order", a.k.a. Big Endian.

Payload Protocol Identifier (PPID) / Fragment Sequence Number (FSN): 32 bits (unsigned integer)
If the B bit is set, this field contains the PPID of the user message. Note that in this case, this field is not touched by an SCTP implementation; therefore, its byte order is not necessarily in network byte order. The upper layer is responsible for any byte order conversions to this field, similar to the PPID of DATA chunks. In this case the FSN is implicitly considered to be 0. If the B bit is not set, this field contains the FSN. The FSN is used to enumerate all fragments of a single user message, starting from 0 and incremented by 1. The last fragment of a message MUST have the E bit set. Note that the FSN MAY wrap completely multiple times allowing arbitrarily large user messages. For the FSN the serial number arithmetic defined in [RFC1982] applies with SERIAL_BITS = 32. Therefore, a sender MUST NOT have more than 2**31 - 1 fragments of a single user message in flight. A fragment is considered in flight, if it is not acknowledged in a non-renegable way. Please note that the FSN is in "network byte order", a.k.a. Big Endian.

2.2. Procedures

This subsection describes how the support of the I-DATA chunk is negotiated and how the I-DATA chunk is used by the sender and receiver.

The handling of the I bit for the I-DATA chunk corresponds to the handling of the I bit for the DATA chunk described in [RFC7053].

2.2.1. Negotiation

An SCTP end point indicates user message interleaving support by listing the I-DATA Chunk within the Supported Extensions Parameter as defined in [RFC5061]. User message interleaving has been negotiated for an association if both end points have indicated I-DATA support.

If user message interleaving support has been negotiated for an association, I-DATA chunks MUST be used for all user messages and DATA-chunks MUST NOT be used. If user message interleaving support has not been negotiated for an association, DATA chunks MUST be used for all user messages and I-DATA chunks MUST NOT be used.

An end point implementing the socket API specified in [RFC6458] MUST NOT indicate user message interleaving support unless the user has requested its use (e.g. via the socket API, see Section 4.3). This constraint is made since the usage of this chunk requires that the application is capable of handling interleaved messages upon reception within an association. This is not the default choice within the socket API (see the SCTP_FRAGMENT_INTERLEAVE socket option in Section 8.1.20 of [RFC6458]) thus the user MUST indicate to the SCTP implementation its support for receiving completely interleaved messages.

Note that stacks that do not implement [RFC6458] may use other methods to indicate interleaved message support and thus indicate the support of user message interleaving. The crucial point is that the SCTP stack MUST know that the application can handle interleaved messages before indicating the I-DATA support.

2.2.2. Sender Side Considerations

The sender side usage of the I-DATA chunk is quite simple. Instead of using the TSN for fragmentation purposes, the sender uses the new FSN field to indicate which fragment number is being sent. The first fragment MUST have the B bit set. The last fragment MUST have the E bit set. All other fragments MUST NOT have the B bit or E bit set. All other properties of the existing SCTP DATA chunk also apply to the I-DATA chunk, i.e. congestion control as well as receiver window conditions MUST be observed as defined in [RFC4960].

Note that the usage of this chunk implies the late assignment of the actual TSN to any chunk being sent. Each I-DATA chunk uses a single TSN. This way messages from other streams may be interleaved with the fragmented message. Please note that this is the only form of interleaving support. For example, it is not possible to interleave multiple ordered or unordered user messages from the same stream.

The sender MUST NOT process (move user data into I-DATA chunks and assign a TSN to it) more than one user message in any given stream at any time. At any time, a sender MAY process multiple user messages, each of them on different streams.

The sender MUST assign TSNs to I-DATA chunks in a way that the receiver can make progress. One way to achieve this is to assign a higher TSN to the later fragments of a user message and send out the I-DATA chunks such that the TSNs are in sequence.

2.2.3. Receiver Side Considerations

Upon reception of an SCTP packet containing an I-DATA chunk whose user message needs to be reassembled, the receiver MUST first use the SID to identify the stream, consider the U bit to determine if it is part of an ordered or unordered message, find the user message identified by the MID and finally use the FSN for reassembly of the message and not the TSN. The receiver MUST NOT make any assumption about the TSN assignments of the sender. Note that a non-fragmented message is indicated by the fact that both the E and B bits are set. A message (either ordered or unordered) may be identified as being fragmented whose E and B bits are not both set.

If I-DATA support has been negotiated for an association, the reception of a DATA chunk is a violation of the above rules and therefore the receiver of the DATA chunk MUST abort the association by sending an ABORT chunk. The ABORT chunk MAY include the ’Protocol Violation’ error cause. The same applies if I-DATA support has not been negotiated for an association and an I-DATA chunk is received.

2.3. Interaction with other SCTP Extensions

The usage of the I-DATA chunk might interfere with other SCTP extensions. Future SCTP extensions MUST describe if and how they interfere with the usage of I-DATA chunks. For the SCTP extensions already defined when this document was published, the details are given in the following subsections.

2.3.1. SCTP Partial Reliability Extension

When the SCTP extension defined in [RFC3758] is used in combination with the user message interleaving extension, the new I-FORWARD-TSN chunk MUST be used instead of the FORWARD-TSN chunk. The difference between the FORWARD-TSN and the I-FORWARD-TSN chunk is that the 16-bit Stream Sequence Number (SSN) has been replaced by the 32-bit Message Identifier (MID) and the largest skipped MID can also be provided for unordered messages. Therefore, the principle applied to

ordered message when using FORWARD-TSN chunks is applied to ordered and unordered messages when using I-FORWARD-TSN chunks.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 194  | Flags = 0x00  |      Length = Variable        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       New Cumulative TSN                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Stream Identifier       |          Reserved           |U|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Message Identifier                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Stream Identifier       |          Reserved           |U|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Message Identifier                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: I-FORWARD-TSN chunk format

The old fields are:

Flags: 8-bits (unsigned integer)
These bits are reserved. They MUST be set to 0 by the sender and MUST be ignored by the receiver. Same as the Flags for FORWARD TSN chunks as specified in [RFC3758].

Length: 16-bits (unsigned integer)
This field holds the length of the chunk. Similar to the Length for FORWARD TSN chunks as specified in [RFC3758].

New Cumulative TSN: 32-bits (unsigned integer)
This indicates the new cumulative TSN to the data receiver. Same as the New Cumulative TSN for FORWARD TSN chunks as specified in [RFC3758].

The new fields are:

Stream Identifier (SID): 16-bits (unsigned integer)
This field holds the stream number this entry refers to.

Reserved: 15 bits
This field is reserved. It MUST be set to 0 by the sender and MUST be ignored by the receiver.

U bit: 1 bit
The U bit specifies if the Message Identifier of this entry refers to unordered messages (U bit is set) or ordered messages (U bit is not set).

Message Identifier (MID): 32 bits (unsigned integer)
This field holds the largest Message Identifier for ordered or unordered messages indicated by the U bit that was skipped for the stream specified by the Stream Identifier. For ordered messages this is similar to the FORWARD-TSN chunk, just replacing the 16-bit SSN by the 32-bit MID.

Support for the I-FORWARD-TSN chunk is negotiated during the SCTP association setup via the Supported Extensions Parameter as defined in [RFC5061]. Only if both end points indicated their support of user message interleaving and the I-FORWARD-TSN chunk, the partial reliability extension is negotiated and can be used in combination with user message interleaving.

The FORWARD-TSN chunk MUST be used in combination with the DATA chunk and MUST NOT be used in combination with the I-DATA chunk. The I-FORWARD-TSN chunk MUST be used in combination with the I-DATA chunk and MUST NOT be used in combination with the DATA chunk.

If I-FORWARD-TSN support has been negotiated for an association, the reception of a FORWARD-TSN chunk is a violation of the above rules and therefore the receiver of the FORWARD-TSN chunk MUST abort the association by sending an ABORT chunk. The ABORT chunk MAY include the ’Protocol Violation’ error cause. The same applies if I-FORWARD-TSN support has not been negotiated for an association and a FORWARD-TSN chunk is received.

2.3.2. SCTP Stream Reconfiguration Extension

When an association resets the SSN using the SCTP extension defined in [RFC6525], the two counters (one for the ordered messages, one for the unordered messages) used for the MIDs MUST be reset to 0.

Since most schedulers, especially all schedulers supporting user message interleaving, require late TSN assignment, it should be noted that the implementation of [RFC6525] needs to handle this.

3. Stream Schedulers

This section defines several stream schedulers. The stream schedulers may behave differently depending on whether user message interleaving has been negotiated for the association or not. An implementation MAY implement any subset of them. If the

implementation is used for WebRTC Datachannels as specified in [I-D.ietf-rtcweb-data-channel] it MUST implement the Weighted Fair Queueing Scheduler defined in Section 3.6.

The selection of the stream scheduler is done at the sender side. There is no mechanism provided for signalling the stream scheduler being used to the receiver side or even let the receiver side influence the selection of the stream scheduler used at the sender side.

3.1. First Come First Served Scheduler (SCTP_SS_FCFS)

The simple first-come, first-served scheduler of user messages is used. It just passes through the messages in the order in which they have been delivered by the application. No modification of the order is done at all. The usage of user message interleaving does not affect the sending of the chunks, except that I-DATA chunks are used instead of DATA chunks.

3.2. Round Robin Scheduler (SCTP_SS_RR)

When not using user message interleaving, this scheduler provides a fair scheduling based on the number of user messages by cycling around non-empty stream queues. When using user message interleaving, this scheduler provides a fair scheduling based on the number of I-DATA chunks by cycling around non-empty stream queues.

3.3. Round Robin Scheduler per Packet (SCTP_SS_RR_PKT)

This is a round-robin scheduler, which only switches streams when starting to fill a new packet. It bundles only DATA or I-DATA chunks referring to the same stream in a packet. This scheduler minimizes head-of-line blocking when a packet is lost because only a single stream is affected.

3.4. Priority Based Scheduler (SCTP_SS_PRIO)

Scheduling of user messages with strict priorities is used. The priority is configurable per outgoing SCTP stream. Streams having a higher priority will be scheduled first and when multiple streams have the same priority, the scheduling between them is implementation dependent. When using user message interleaving, the sending of large lower priority user messages will not delay the sending of higher priority user messages.

3.5. Fair Capacity Scheduler (SCTP_SS_FC)

A fair capacity distribution between the streams is used. This scheduler considers the lengths of the messages of each stream and schedules them in a specific way to maintain an equal capacity for all streams. The details are implementation dependent. Using user message interleaving allows for a better realization of the fair capacity usage.

3.6. Weighted Fair Queueing Scheduler (SCTP_SS_WFQ)

A weighted fair queueing scheduler between the streams is used. The weight is configurable per outgoing SCTP stream. This scheduler considers the lengths of the messages of each stream and schedules them in a specific way to use the capacity according to the given weights. If the weight of stream S1 is n times the weight of stream S2, the scheduler should assign to stream S1 n times the capacity it assigns to stream S2. The details are implementation dependent. Using user message interleaving allows for a better realization of the capacity usage according to the given weights.

This scheduler in combination with user message interleaving is used for WebRTC Datachannels as specified in [I-D.ietf-rtcweb-data-channel].

4. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to allow applications to use the extension described in this document.

Please note that this section is informational only.

4.1. Exposure of the Stream Sequence Number (SSN)

The socket API defined in [RFC6458] defines several structures in which the SSN of a received user message is exposed to the application. The list of these structures includes:

struct sctp_sndrcvinfo
: Specified in Section 5.3.2 SCTP Header Information Structure (SCTP_SNDRCV) of [RFC6458] and marked as deprecated.

struct sctp_extrcvinfo
: Specified in Section 5.3.3 Extended SCTP Header Information Structure (SCTP_EXTRCV)of [RFC6458] and marked as deprecated.

struct sctp_rcvinfo

Specified in Section 5.3.5 SCTP Receive Information Structure (SCTP_RCVINFO) of [RFC6458].

If user message interleaving is used, the lower order 16 bits of the MID are used as the SSN when filling out these structures.

4.2. SCTP_ASSOC_CHANGE Notification

When an SCTP_ASSOC_CHANGE notification (specified in Section 6.1.1 of [RFC6458]) is delivered indicating a sac_state of SCTP_COMM_UP or SCTP_RESTART for an SCTP association where both peers support the I-DATA chunk, SCTP_ASSOC_SUPPORTS_INTERLEAVING should be listed in the sac_info field.

4.3. Socket Options

option name	data type	get	set
SCTP_INTERLEAVING_SUPPORTED	struct sctp_assoc_value	X	X
SCTP_STREAM_SCHEDULER	struct sctp_assoc_value	X	X
SCTP_STREAM_SCHEDULER_VALUE	struct sctp_stream_value	X	X

4.3.1. Enable or Disable the Support of User Message Interleaving (SCTP_INTERLEAVING_SUPPORTED)

This socket option allows the enabling or disabling of the negotiation of user message interleaving support for future associations. For existing associations it allows to query whether user message interleaving support was negotiated or not on a particular association.

This socket option uses IPPROTO_SCTP as its level and SCTP_INTERLEAVING_SUPPORTED as its name. It can be used with getsockopt() and setsockopt(). The socket option value uses the following structure defined in [RFC6458]:

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets, this parameter indicates upon which association the user is performing an action. The special

sctp_assoc_t SCTP_FUTURE_ASSOC can also be used, it is an error to use SCTP_{CURRENT|ALL}_ASSOC in assoc_id.

assoc_value: A non-zero value encodes the enabling of user message interleaving whereas a value of 0 encodes the disabling of user message interleaving.

sctp_opt_info() needs to be extended to support SCTP_INTERLEAVING_SUPPORTED.

An application using user message interleaving should also set the fragment interleave level to 2 by using the SCTP_FRAGMENT_INTERLEAVE socket option specified in Section 8.1.20 of [RFC6458]. This allows the interleaving of user messages from different streams. Please note that it does not allow the interleaving of user messages (ordered or unordered) on the same stream. Failure to set this option can possibly lead to application deadlock. Some implementations might therefore put some restrictions on setting combinations of these values. Setting the interleaving level to at least 2 before enabling the negotiation of user message interleaving should work on all platforms. Since the default fragment interleave level is not 2, user message interleaving is disabled per default.

4.3.2. Get or Set the Stream Scheduler (SCTP_STREAM_SCHEDULER)

A stream scheduler can be selected with the SCTP_STREAM_SCHEDULER option for setsockopt(). The struct sctp_assoc_value is used to specify the association for which the scheduler should be changed and the value of the desired algorithm.

The definition of struct sctp_assoc_value is the same as in [RFC6458]:

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id: Holds the identifier for the association of which the scheduler should be changed. The special SCTP_{FUTURE|CURRENT|ALL}_ASSOC can also be used. This parameter is ignored for one-to-one style sockets.

assoc_value: This specifies which scheduler is used. The following constants can be used:

SCTP_SS_DEFAULT: The default scheduler used by the SCTP implementation. Typical values are SCTP_SS_FCFS or SCTP_SS_RR.

SCTP_SS_FCFS: Use the scheduler specified in Section 3.1.

SCTP_SS_RR: Use the scheduler specified in Section 3.2.

SCTP_SS_RR_PKT: Use the scheduler specified in Section 3.3.

SCTP_SS_PRIO: Use the scheduler specified in Section 3.4. The priority can be assigned with the sctp_stream_value struct. The higher the assigned value, the lower the priority, that is the default value 0 is the highest priority and therefore the default scheduling will be used if no priorities have been assigned.

SCTP_SS_FB: Use the scheduler specified in Section 3.5.

SCTP_SS_WFQ: Use the scheduler specified in Section 3.6. The weight can be assigned with the sctp_stream_value struct.

sctp_opt_info() needs to be extended to support SCTP_STREAM_SCHEDULER.

4.3.3. Get or Set the Stream Scheduler Parameter (SCTP_STREAM_SCHEDULER_VALUE)

Some schedulers require additional information to be set for individual streams as shown in the following table:

name	per stream info
SCTP_SS_DEFAULT	n/a
SCTP_SS_FCFS	no
SCTP_SS_RR	no
SCTP_SS_RR_PKT	no
SCTP_SS_PRIO	yes
SCTP_SS_FB	no
SCTP_SS_WFQ	yes

This is achieved with the SCTP_STREAM_SCHEDULER_VALUE option and the corresponding struct sctp_stream_value. The definition of struct sctp_stream_value is as follows:

```
struct sctp_stream_value {
  sctp_assoc_t assoc_id;
  uint16_t stream_id;
  uint16_t stream_value;
};
```

assoc_id: Holds the identifier for the association of which the scheduler should be changed. The special SCTP_{FUTURE|CURRENT|ALL}_ASSOC can also be used. This parameter is ignored for one-to-one style sockets.

stream_id: Holds the stream id of the stream for which additional information has to be provided.

stream_value: The meaning of this field depends on the scheduler specified. It is ignored when the scheduler does not need additional information.

sctp_opt_info() needs to be extended to support SCTP_STREAM_SCHEDULER_VALUE.

4.4. Explicit EOR Marking

Using explicit End of Record (EOR) marking for an SCTP association supporting user message interleaving allows the user to interleave the sending of user messages on different streams.

5. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

[NOTE to RFC-Editor:

The suggested values for the chunk types and the chunk flags are tentative and to be confirmed by IANA.

]

This document (RFCXXXX) is the reference for all registrations described in this section.

Two new chunk types have to be assigned by IANA.

5.1. I-DATA Chunk

IANA should assign the chunk type for this chunk from the pool of chunks with the upper two bits set to ’01’. This requires an additional line in the "Chunk Types" registry for SCTP:

ID Value	Chunk Type	Reference
64	Payload Data supporting Interleaving (I-DATA)	[RFCXXXX]

The registration table as defined in [RFC6096] for the chunk flags of this chunk type is initially given by the following table:

Chunk Flag Value	Chunk Flag Name	Reference
0x01	E bit	[RFCXXXX]
0x02	B bit	[RFCXXXX]
0x04	U bit	[RFCXXXX]
0x08	I bit	[RFCXXXX]
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

5.2. I-FORWARD-TSN Chunk

IANA should assign the chunk type for this chunk from the pool of chunks with the upper two bits set to ’11’. This requires an additional line in the "Chunk Types" registry for SCTP:

ID Value	Chunk Type	Reference
194	I-FORWARD-TSN	[RFCXXXX]

The registration table as defined in [RFC6096] for the chunk flags of this chunk type is initially empty.

6. Security Considerations

This document does not add any additional security considerations in addition to the ones given in [RFC4960] and [RFC6458].

It should be noted that the application has to consent that it is willing to do the more complex reassembly support required for user message interleaving. When doing so, an application has to provide a reassembly buffer for each incoming stream. It has to protect itself against these buffers taking too many resources. If user message

interleaving is not used, only a single reassembly buffer needs to be provided for each association. But the application has to protect itself for excessive resource usages there too.

7. Acknowledgments

The authors wish to thank Benoit Claise, Julian Cordes, Spencer Dawkins, Gorry Fairhurst, Lennart Grahl, Christer Holmberg, Mirja Kuehlewind, Marcelo Ricardo Leitner, Karen E. Egede Nielsen, Maksim Proshin, Eric Rescorla, Irene Ruengeler, Felix Weinrank, Michael Welzl, Magnus Westerlund, and Lixia Zhang for their invaluable comments.

This work has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 644334 (NEAT). The views expressed are solely those of the author(s).

8. References

8.1. Normative References

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

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

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

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

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <https://www.rfc-editor.org/info/rfc5061>.

[RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, DOI 10.17487/RFC6096, January 2011, <https://www.rfc-editor.org/info/rfc6096>.

[RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, DOI 10.17487/RFC6525, February 2012, <https://www.rfc-editor.org/info/rfc6525>.

[RFC7053] Tuexen, M., Ruengeler, I., and R. Stewart, "SACK-IMMEDIATELY Extension for the Stream Control Transmission Protocol", RFC 7053, DOI 10.17487/RFC7053, November 2013, <https://www.rfc-editor.org/info/rfc7053>.

8.2. Informative References

[I-D.ietf-rtcweb-data-channel] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

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

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <https://www.rfc-editor.org/info/rfc6458>.

Authors' Addresses

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

Email: randall@lakerest.net

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

Email: tuexen@fh-muenster.de

Salvatore Loreto
Ericsson
Torshamnsgatan 21
164 80 Stockholm
Sweden

Email: Salvatore.Loreto@ericsson.com

Robin Seggelmann
Metafinanz Informationssysteme GmbH
Leopoldstrasse 146
80804 Muenchen
Germany

Email: rfc@robin-seggelmann.com

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

Copyright Notice

Table of Contents

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:

pr_policy	pr_value	Specification
SCTP_PR_SCTP_NONE	Ignored	[RFC6458]
SCTP_PR_SCTP_TTL	Lifetime in ms	[RFC6458]
SCTP_PR_SCTP_RTX	Number of retransmissions	Section 3.1
SCTP_PR_SCTP_PRIO	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, Gorry 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

RTGWG
Internet-Draft
Intended status: Informational
Expires: November 22, 2015

E. Nordmark (ed)
Arista Networks
A. Tian
Ericsson Inc.
J. Gross
VMware
J. Hudson
Brocade Communications Systems, Inc.
L. Kreeger
Cisco Systems, Inc.
P. Garg
Microsoft
P. Thaler
Broadcom Corporation
T. Herbert
Google
May 21, 2015

Encapsulation Considerations
draft-rtg-dt-encap-02

Abstract

The IETF Routing Area director has chartered a design team to look at common issues for the different data plane encapsulations being discussed in the NVO3 and SFC working groups and also in the BIER BoF, and also to look at the relationship between such encapsulations in the case that they might be used at the same time. The purpose of this design team is to discover, discuss and document considerations across the different encapsulations in the different WGs/BoFs so that we can reduce the number of wheels that need to be reinvented in the future.

Status of this Memo

material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 22, 2015.

Copyright Notice

Table of Contents

1. Design Team Charter

There have been multiple efforts over the years that have resulted in new or modified data plane behaviors involving encapsulations. That includes IETF efforts like MPLS, LISP, and TRILL but also industry efforts like VXLAN and NVGRE. These collectively can be seen as a source of insight into the properties that data planes need to meet. The IETF is currently working on potentially new encapsulations in NVO3 and SFC and considering working on BIER. In addition there is work on tunneling in the INT area.

This is a short term design team chartered to collect and construct useful advice to parties working on new or modified data plane behaviors that include additional encapsulations. The goal is for the group to document useful advice gathered from interacting with ongoing efforts. An Internet Draft will be produced for IETF92 to capture that advice, which will be discussed in RTGWG.

Data plane encapsulations face a set of common issues such as:

- o How to provide entropy for ECMP
- o Issues around packet size and fragmentation/reassembly
- o OAM - what support is needed in an encapsulation format?
- o Security and privacy.
- o QoS
- o Congestion Considerations
- o IPv6 header protection (zero UDP checksum over IPv6 issue)
- o Extensibility - e.g., for evolving OAM, security, and/or congestion control
- o Layering of multiple encapsulations e.g., SFC over NVO3 over BIER

The design team will provide advice on those issues. The intention is that even where we have different encapsulations for different purposes carrying different information, each such encapsulation doesn’t have to reinvent the wheel for the above common issues.

The design team will look across the routing area in particular at SFC, NVO3 and BIER. It will not be involved in comparing or analyzing any particular encapsulation formats proposed in those WGs and BoFs but instead focus on common advice.

2. Overview

The references provide background information on NVO3, SFC, and BIER. In particular, NVO3 is introduced in [RFC7364], [RFC7365], and [I-D.ietf-nvo3-arch]. SFC is introduced in [I-D.ietf-sfc-architecture] and [I-D.ietf-sfc-problem-statement]. Finally, the information on BIER is in [I-D.shepherd-bier-problem-statement],

[I-D.wijnands-bier-architecture], and [I-D.wijnands-mpls-bier-encapsulation]. We assume the reader has some basic familiarity with those proposed encapsulations. The Related Work section points at some prior work that relates to the encapsulation considerations in this document.

Encapsulation protocols typically have some unique information that they need to carry. In some cases that information might be modified along the path and in other cases it is constant. The in-flight modifications has impacts on what it means to provide security for the encapsulation headers.

- o NVO3 carries a VNI Identifier edge to edge which is not modified. There has been OAM discussions in the WG and it isn’t clear whether some of the OAM information might be modified in flight.
- o SFC carries service meta-data which might be modified or unmodified as the packets follow the service path. SFC talks of some loop avoidance mechanism which is likely to result in modifications for for each hop in the service chain even if the meta-data is unmodified.
- o BIER carries a bitmap of egress ports to which a packet should be delivered, and as the packet is forwarded down different paths different bits are cleared in that bitmap.

Even if information isn’t modified in flight there might be devices that wish to inspect that information. For instance, one can envision future NVO3 security devices which filter based on the virtual network identifier.

The need for extensibility is different across the protocols

- o NVO3 might need some extensions for OAM and security.
- o SFC is all about carrying service meta-data along a path, and different services might need different types and amount of meta-data.
- o BIER might need variable number of bits in their bitmaps, or other future schemes to scale up to larger network.

The extensibility needs and constraints might be different when considering hardware vs. software implementations of the encapsulation headers. NIC hardware might have different constraints than switch hardware.

As the IETF designs these encapsulations the different WGs solve the issues for their own encapsulation. But there are likely to be future cases when the different encapsulations are combined in the same header. For instance, NVO3 might be a "transport" used to carry SFC between the different hops in the service chain.

Most of the issues discussed in this document are not new. The IETF and industry as specified and deployed many different encapsulation

or tunneling protocols over time, ranging from simple IP-in-IP and GRE encapsulation, IPsec, pseudo-wires, session-based approached like L2TP, and the use of MPLS control and data planes. IEEE 802 has also defined layered encapsulation for Provider Backbone Bridges (PBB) and IEEE 802.1Qbp (ECMP). This document tries to leverage what we collectively have learned from that experience and summarize what would be relevant for new encapsulations like NVO3, SFC, and BIER.

3. Common Issues

[This section is mostly a repeat of the charter but with a few modifications and additions.]

Any new encapsulation protocol would need to address a large set of issues that are not central to the new information that this protocol intends to carry. The common issues explored in this document are:

- How to provide entropy for Equal Cost MultiPath (ECMP) routing
- Issues around packet size and fragmentation/reassembly
- Next header indication - each encapsulation might be able to carry different payloads
- OAM - what support is needed in an encapsulation format?
- Security and privacy
- QoS
- Congestion Considerations
- Header protection
- Extensibility - e.g., for evolving OAM, security, and/or congestion control
- Layering of multiple encapsulations e.g., SFC over NVO3 over BIER
- Importance of being friendly to hardware and software implementations

The degree to which these common issues apply to a particular encapsulation can differ based on the intended purpose of the encapsulation. But it is useful to understand all of them before determining which ones apply.

4. Scope

It is important to keep in mind what we are trying to cover and not cover in this document and effort. This is

- A look across the three new encapsulations, while taking lots of previous work into account
- Focus on the class of encapsulations that would run over IP/UDP. That was done to avoid being distracted by the data-plane and control-plane interaction, which is more significant for protocols that are designed to run over "transports" that maintain session

or path state.
- o We later expanded the scope somewhat to consider how the encapsulations would play with MPLS "transport", which is important because SFC and BIER seem to target being independent of the underlying "transport"

However, this document and effort is NOT intended to:
- o Design some new encapsulation header to rule them all
- o Design yet another new NVO3 encapsulation header
- o Try to select the best encapsulation header
- o Evaluate any existing and proposed encapsulations

While the origin and focus of this document is the routing area and in particular NVO3, SFC, and BIER, the considerations apply to other encapsulations that are being defined in the IETF and elsewhere. There seems to be an increase in the number of encapsulations being defined to run over UDP, where there might already exist an encapsulation over IP or Ethernet. Feedback on how these considerations apply in those contexts is welcome.

5. Assumptions

The design center for the new encapsulations is a well-managed network. That network can be a datacenter network (plus datacenter interconnect) or a service provider network. Based on the existing and proposed encapsulations in those environment it is reasonable to make these assumptions:
- o The MTU is carefully managed and configured. Hence an encapsulation protocol can make the packets bigger without resulting in a requirement for fragmentation and reassembly between ingress and egress. (However, it might be useful to detecting MTU misconfigurations.)
- o In general an encapsulation needs some approach for congestion management. But the assumptions are different than for arbitrary Internet paths in that the underlay might be well-provisioned and better policed at the edge, and due to multi-tenancy, the congestion control in the endpoints might be even less trusted than on the Internet at large.

The goal is to implement these encapsulations in hardware and software hence we can’t assume that the needs of either implementation approach can trump the needs of the other. In particular, around extensibility the needs and constraints might be quite different.

6. Terminology

The capitalized keyword MUST is used as defined in http://en.wikipedia.org/wiki/Julmust

TBD: Refer to existing documents for at least NVO3 and SFC terminology. We use at least the VNI ID in this document.

7. Entropy

In many cases the encapsulation format needs to enable ECMP in unmodified routers. Those routers might use different fields in TCP/UDP packets to do ECMP without a risk of reordering a flow.

The common way to do ECMP-enabled encapsulation over IP today is to add a UDP header and to use UDP with the UDP source port carrying entropy from the inner/original packet headers as in LISP [RFC6830]. The total entropy consists of 14 bits in the UDP source port (using the ephemeral port range) plus the outer IP addresses which seems to be sufficient for entropy; using outer IPv6 headers would give the option for more entropy should it be needed in the future.

In some environments it might be fine to use all 16 bits of the port range. However, middleboxes might make assumptions about the system ports or user ports. But they should not make any assumptions about the ports in the Dynamic and/or Private Port range, which have the two MSBs set to 11b.

The UDP source port might change over the lifetime of an encapsulated flow, for instance for DoS mitigation or re-balancing load across ECMP.

There is some interaction between entropy and OAM and extensibility mechanism. It is desirable to be able to send OAM packets to follow the same path as network packets. Hence OAM packets should use the same entropy mechanism as data packets. While routers might use information in addition the entropy field and outer IP header, they can not use arbitrary parts of the encapsulation header since that might result in OAM frames taking a different path. Likewise if routers look past the encapsulation header they need to be aware of the extensibility mechanism(s) in the encapsulation format to be able to find the inner headers in the presence of extensions; OAM frames might use some extensions e.g. for timestamps.

Architecturally the entropy and the next header field are really part of enclosing delivery header. UDP with entropy goes hand-in-hand with the outer IP header. Thus the UDP entropy is present for the

underlay IP routers the same way that an MPLS entropy label is present for LSRs. The entropy above is all about providing entropy for the outer delivery of the encapsulated packets.

It has been suggested that when IPv6 is used it would not be necessary to add a UDP header for entropy, since the IPv6 flow label can be used for entropy. (This assumes that there is an IP protocol number for the encapsulation in addition to a UDP destination port number since UDP would be used with IPv4 underlay. And any use of UDP checksums would need to be replaced by an encaps-specific checksum or secure hash.) While such an approach would save 8 bytes of headers when the underlay is IPv6, it does assume that the underlay routers use the flow label for ECMP, and it also would make the IPv6 approach different than the IPv4 approach. Currently the leaning is towards recommending using the UDP encapsulation for both IPv4 and IPv6 underlay. The IPv6 flow label can be used for additional entropy if need be.

Note that in the proposed BIER encapsulation [I-D.wijnands-mpls-bier-encapsulation], there is an an 8-bit field which specifies an entropy value that can be used for load balancing purposes. This entropy is for the BIER forwarding decisions, which is independent of any outer delivery ECMP between BIER routers. Thus it is not part of the delivery ECMP discussed in this section.

[Note: For any given bit in BIER (that identifies an exit from the BIER domain) there might be multiple immediate next hops. The BIER entropy field is used to select that next hop as part of BIER processing. The BIER forwarding process may do equal cost load balancing, but the load balancing procedure MUST choose the same path for any two packets have the same entropy value.]

In summary:

- o The entropy is associated with the transport, that is an outer IP header or MPLS.
- o In the case of IP transport use >=14 bits of UDP source port, plus outer IPv6 flowid for entropy.

8. Next-protocol indication

Next-protocol indications appear in three different context for encapsulations.

Firstly, the transport delivery mechanism for the encapsulations we discuss in this document need some way to indicate which encapsulation header (or other payload) comes next in the packet. Some encapsulations might be identified by a UDP port; others might be identified by an Ethernet type or IP protocol number. Which

approach is used is a function of the preceding header the same way as IPv4 is identified by both an Ethernet type and an IP protocol number (for IP-in-IP). In some cases the header type is implicit in some session (L2TP) or path (MPLS) setup. But this is largely beyond the control of the encapsulation protocol. For instance, if there is a requirement to carry the encapsulation after an Ethernet header, then an Ethernet type is needed. If required to be carried after an IP/UDP header, then a UDP port number is needed. For UDP port numbers there are considerations for port number conservation described in [I-D.ietf-tsvwg-port-use].

It is worth mentioning that in the MPLS case of no implicit protocol type many forwarding devices peek at the first nibble of the payload to determine whether to apply IPv4 or IPv6 L3/L4 hashes for load balancing [RFC7325]. That behavior places some constraints on other payloads carried over MPLS and some protocol define an initial control word in the payload with a value of zero in its first nibble [RFC4385] to avoid confusion with IPv4 and IPv6 payload headers.

Secondly, the encapsulation needs to indicate the type of its payload, which is in scope for the design of the encapsulation. We have existing protocols which use Ethernet types (such as GRE). Here each encapsulation header can potentially makes its own choices between:

- o Reuse Ethernet types - makes it easy to carry existing L2 and L3 protocols including IPv6, IPv6, and Ethernet. Disadvantages are that it is a 16 bit number and we probably need far less than 100 values, and the number space is controlled by the IEEE 802 RAC with its own allocation policies.
- o Reuse IP protocol numbers - makes it easy to carry e.g., ESP in addition to IP and Etnernet but brings in all existing protocol numbers many of which would never be used directly on top of the encapsulation protocol. IANA managed eight bit values, presumably more difficult to get an assigned number than to get a transport port assignment.
- o Define their own next-protocol number space, which can use fewer bits than an Ethernet type and give more flexibility, but at the cost of administering that numbering space (presumably by the IANA).

Thirdly, if the IETF ends up defining multiple encapsulations at about the same time, and there is some chance that multiple such encapsulations can be combined in the same packet, there is a question whether it makes sense to use a common approach and numbering space for the encapsulation across the different protocols. A common approach might not be beneficial as long as there is only one way to indicate e.g., SFC inside NVO3.

Many Internet protocols use fixed values (typically managed by the IANA function) for their next-protocol field. That facilitates interpretation of packets by middleboxes and e.g., for debugging purposes, but might make the protocol evolution inflexible. Our collective experience with MPLS shows an alternative where the label can be viewed as an index to a table containing processing instructions and the table content can be managed in different ways. Encapsulations might want to consider the tradeoffs between such more flexible versus more fixed approaches.

In summary:

- o Would it be useful for the IETF come up with a common scheme for encapsulation protocols? If not each encapsulation can define its own scheme.

9. MTU and Fragmentation

A common approach today is to assume that the underlay have sufficient MTU to carry the encapsulated packets without any fragmentation and reassembly at the tunnel endpoints. That is sufficient when the operator of the ingress and egress have full control of the paths between those endpoints. And it makes for simpler (hardware) implementations if fragmentation and reassembly can be avoided.

However, even under that assumption it would be beneficial to be able to detect when there is some misconfiguration causing packets to be dropped due to MTU issues. One way to do this is to have the encapsulator set the don’t-fragment (DF) flag in the outer IPv4 header and receive and log any received ICMP "packet too big" (PTB) errors. Note that no flag needs to be set in an outer IPv6 header [RFC2460].

Encapsulations could also define an optional tunnel fragmentation and reassembly mechanism which would be useful in the case when the operator doesn’t have full control of the path, or when the protocol gets deployed outside of its original intended context. Such a mechanism would be required if the underlay might have a path MTU which makes it impossible to carry at least 1518 bytes (if offering Ethernet service), or at least 1280 (if offering IPv6 service). The use of such a protocol mechanism could be triggered by receiving a PTB. But such a mechanism might not be implemented by all encapsulators and decapsulators. [Aerolink is one example of such a protocol.]

Depending on the payload carried by the encapsulation there are some additional possibilities:

- o If payload is IPv4/6 then the underlay path MTU could be used to report end-to-end path MTU.
- o If the payload service is Ethernet/L2, then there is no such per destination reporting mechanism. However, there is a LLDP TLV for reporting max frame size; might be useful to report minimum to end stations, but unmodified end stations would do nothing with that TLV since they assume that the MTU is at least 1518.

In summary:

- o In some deployments an encapsulation can assume well-managed MTU hence no need for fragmentation and reassembly related to the encapsulation.
- o Even so, it makes sense for ingress to track any ICMP packet too big addressed to ingress to be able to log any MTU misconfigurations.
- o Should an encapsulation protocol be depoyed outside of the original context it might very well need support for fragmentation and reassembly.

10. OAM

The OAM area is seeing active development in the IETF with discussions (at least) in NVO3 and SFC working groups, plus the new LIME WG looking at architecture and YANG models.

The design team has take a narrow view of OAM to explore the potential OAM implications on the encapsulation format.

In terms of what we have heard from the various working groups there seem to be needs to:

- o Be able to send out-of-band OAM messages - that potentially should follow the same path through the network as some flow of data packets.
  - * Such OAM messages should not accidentally be decapsulated and forwarded to the end stations.
  - * Be able to add OAM information to data packets that are encapsulated. Discussions have been around
  - * Using a bit in the OAM to synchronize sampling of counters between the encapsulator and decapsulator.
  - * Optional timestamps, sequence numbers, etc for more detailed measurements between encapsulator and decapsulator.
- o Usable for both proactive monitoring (akin to BFD) and reactive checks (akin to traceroute to pin-point a failure)

To ensure that the OAM messages can follow the same path the OAM messages need to get the same ECMP (and LAG hashing) results as a given data flow. An encapsulator can choose between one of:

- o Limit ECMP hashing to not look past the UDP header i.e. the entropy needs to be in the source/destination IP and UDP ports
- o Make OAM packets look the same as data packets i.e. the initial part of the OAM payload has the inner Ethernet, IP, TCP/UDP headers as a payload. (This approach was taken in TRILL out of necessity since there is no UDP header.) Any OAM bit in the encapsulation header must in any case be excluded from the entropy.

There can be several ways to prevent OAM packets from accidentally being forwarded to the end station using:

- o A bit in the frame (as in TRILL) indicating OAM
- o A next-protocol indication with a designated value for "none" or "oam".

This assumes that the bit or next protocol, respectively, would not affect entropy/ECMP in the underlay. However, the next-protocol field might be used to provide differentiated treatement of packets based on their payload; for instance a TCP vs. IPsec ESP payload might be handled differently. Based on that observation it might be undesirable to overload the next protocol with the OAM drop behavior, resulting in a preference for having a bit to indicate that the packet should be forwarded to the end station after decapsulation.

There has been suggestions that one (or more) marker bits in the encaps header would be useful in order to delineate measurement epochs on the encapsulator and decapsulator and use that to compare counters to determine packet loss.

A result of the above is that OAM is likely to evolve and needs some degree of extensibility from the encapsulation format; a bit or two plus the ability to define additional larger extensions.

An open question is how to handle error messages or other reports relating to OAM. One can think if such reporting as being associated with the encapsulation the same way ICMP is associated with IP. Would it make sense for the IETF to develop a common Encapsulation Error Reporting Protocol as part of OAM, which can be used for different encapsulations? And if so, what are the technical challenges. For instance, how to avoid it being filtered as ICMP often is?

A potential additional consideration for OAM is the possible future existence of gateways that "stitch" together different dataplane encapsulations and might want to carry OAM end-to-end across the different encapsulations.

In summary:

- o It makes sense to reserve a bit for "drop after decapsulation" for OAM out-of-band.
- o An encapsulation needs sufficient extensibility for OAM (such as bits, timestamps, sequence numbers). That might be motivated by in-band OAM but it would make sense to leverage the same extensions for out-of band OAM.
- o OAM places some constraints on use of entropy in forwarding devices.
- o Should IETF look into error reporting that is independent of the specific encapsulation?

11. Security Considerations

Different encapsulation use cases will have different requirements around security. For instance, when encapsulation is used to build overlay networks for network virtualization, isolation between virtual networks may be paramount. BIER support of multicast may entail different security requirements than encapsulation for unicast.

In real deployment, the security of the underlying network may be considered for determining the level of security needed in the encapsulation layer. However for the purposes of this discussion, we assume that network security is out of scope and that the underlying network does not itself provide adequate or as least uniform security mechanisms for encapsulation.

There are at least three considerations for security:
- o Anti-spoofing/virtual network isolation
- o Interaction with packet level security such as IPsec or DTLS
- o Privacy (e.g., VNI ID confidentially for NVO3)

This section uses a VNI ID in NVO3 as an example. A SFC or BIER encapsulation is likely to have fields with similar security and privacy requirements.

11.1. Encapsulation-specific considerations

Some of these considerations appear for a new encapsulation, and others are more specific to network virtualization in datacenters.
- o New attack vectors:
  - * DDOS on specific queued/paths by attempting to reproduce the 5-tuple hash for targeted connections.
  - * Entropy in outer 5-tuple may be too little or predictable.
  - * Leakage of identifying information in the encapsulation header for an encrypted payload.

    * Vulnerabilities of using global values in fields like VNI ID.
  o Trusted versus untrusted tenants in network virtualization:
    * The criticality of virtual network isolation depends on whether tenants are trusted or untrusted. In the most extreme cases, tenants might not only be untrusted but may be considered hostile.
    * For a trusted set of users (e.g. a private cloud) it may be sufficient to have just a virtual network identifier to provide isolation. Packets inadvertently crossing virtual networks should be dropped similar to a TCP packet with a corrupted port being received on the wrong connection.
    * In the presence of untrusted users (e.g. a public cloud) the virtual network identifier must be adequately protected against corruption and verified for integrity. This case may warrant keyed integrity.
  o Different forms of isolation:
    * Isolation could be blocking all traffic between tenants (or except as allowed by some firewall)
    * Could also be about performance isolation i.e. one tenant can overload the network in a way that affects other tenants
    * Physical isolation of traffic for different tenants in network may be required, as well as required restrictions that tenants may have on where their packets may be routed.
  o New attack vectors from untrusted tenants:
    * Third party VMs with untrusted tenants allows internally borne attacks within data centers
    * Hostile VMs inside the system may exist (e.g. public cloud)
    * Internally launched DDOS
    * Passive snooping for mis-delivered packets
    * Mitigate damage and detection in event that a VM is able to circumvent isolation mechanisms
  o Tenant-provider relationship:
    * Tenant might not trust provider, hypervisors, network
    * Provider likely will need to provide SLA or a least a statement on security
    * Tenant may implement their own additional layers of security
    * Regulation and certification consuderations
  o Trend towards tighter security:
    * Tenants’ data in network increases in volume and value, attacks become more sophisticated
    * Large DCs already encrypt everything on disk
    * DCs likely to encrypt inter-DC traffic at this point, use TLS to Internet.
    * Encryption within DC is becoming more commonplace, becomes ubiquitous when cost is low enough.
    * Cost/performance considerations. Cost of support for strong security has made strong network security in DCs prohibitive.

* Are there lessons from MacSec?

11.2. Virtual network isolation

The first requirement is isolation between virtual networks. Packets sent in one virtual network should never be illegitimately received by a node in another virtual network. Isolation should be protected in the presence of malicious attacks or inadvertent packet corruption.

The second requirement is sender authentication. Sender identity is authenticated to prevent anti-spoofing. Even if an attacker has access to the packets in the network, they cannot send packets into a virtual network. This may have two possibilities:

- o Pairwise sender authentication. Any two communicating hosts negotiate a shared key.
- o Group authentication. A group of hosts share a key (this may be more appropriate for multicast of encapsulation).

Possible security solutions:

- o Security cookie: This is similar to L2TP cookie mechanism [RFC3931]. A shared plain text cookie is shared between encapsulator and decapsulator. A receiver validates a packet by evaluating if the cookie is correct for the virtual network and address of a sender. Validation function is F(cookie, VNI ID, source addr). If cookie matches, accept packet, else drop. Since cookie is plain text this method does not protect against an eavesdropping. Cookies are set and may be rotated out of band.
- o Secure hash: This is a stronger mechanism than simple cookies that borrows from IPsec and PPP authentication methods. In this model security field contains a secure hash of some fields in the packet using a shared key. Hash function may be something like H(key, VNI ID, addrs, salt). The salt ensures the hash is not the same for every packet, and if it includes a sequence number may also protect against replay attacks.

In any use of a shared key, periodic re-keying should be allowed. This could include use of techniques like generation numbers, key windows, etc. See [I-D.farrelll-mpls-opportunistic-encrypt] for an example application.

We might see firewalls that are aware of the encapsulation and can provide some defense in depth combined with the above example anti-spoofing approaches. An example would be an NVO3-aware firewall being able to check the VNI ID.

Separately and in addition to such filtering, there might be a desire to completely block an encapsulation protocol at certain places in

the network, e.g., at the edge of a datacenter. Using a fixed standard UDP destination port number for each encapsulation protocol would facilitate such blocking.

11.3. Packet level security

An encapsulated packet may itself be encapsulated in IPsec (e.g. ESP). This should be straightforward and in fact is what would happen today in security gateways. In this case, there is no special consideration for the fact that packet is encapsulated, however since the encapsulation layer headers are included (part of encrypted data for instance) we lose visibility in the network of the encapsulation.

The more interesting case is when security is applied to the encapsulation payload. This will keep the encapsulation headers in the outer header visible to the network (for instance in nvo3 we may way to firewall based on VNI ID even if the payload is encrypted). One possibility is to apply DTLS to the encapsulation payload. In this model the protocol stack may be something like IP|UDP|Encap|DTLS|encrypted_payload. The encapsulation and security should be done together at an encapsulator and resolved at the decapsulator. Since the encapsulation header is outside of the security coverage, this may itself require security (like described above).

In both of the above the security associations (SAs) may be between physical hosts, so for instance in nvo3 we can have packets of different virtual networks using the same SA-- this should not be an issue since it is the VNI ID that ensures isolation (which needs to be secured also).

11.4. In summary:

- o Encapsulations need extensibility mechanisms to be able to add security features like cookies and secure hashes protecting the encapsulation header.
- o NVO3 proably has specific higher requirements relating to isolation for network virtualization, which is in scope for the NVO3 WG/
- o Our collective IETF experience is that succesful protocols get deployed outside of the original intended context, hence the initial assumptions about the threat model might become invalid. That needs to be considered in the standardization of new encapsulations.

12. QoS

In the Internet architecture we support QoS using the Differentiated

Services Code Points (DSCP) in the formerly named Type-of-Service field in the IPv4 header, and in the Traffic-Class field in the IPv6 header. The ToS and TC fields also contain the two ECN bits.

We have existing specifications how to process those bits. See [RFC2983] for diffserv handling, which specifies how the received DSCP value is used to set the DSCP value in an outer IP header when encapsulating. (There are also existing specifications how DSCP can be mapped to layer2 priorities.)

Those specifications apply whether or not there is some intervening headers (e.g., for NVO3 or SFC) between the inner and outer IP headers. Thus the encapsulation considerations in this area are mainly about applying the framework in [RFC2983].

Note that the DSCP and ECN bits are not the only part of an inner packet that might potentially affect the outer packet. For example, [RFC2473] specifies handling of inner IPv6 hop-by-hop options that effectively result in copying some options to the outer header. It is simpler to not have future encapsulations depend on such copying behavior.

There are some other considerations specific to doing OAM for encapsulations. If OAM messages are used to measure latency, it would make sense to treat them the same as data payloads. Thus they need to have the same outer DSCP value as the data packets which they wish to measure.

Due to OAM there are constraints on middleboxes in general. If middleboxes inspect the packet past the outer IP+UDP and encapsulation header and look for inner IP and TCP/UDP headers, that might violate the assumption that OAM packets will be handled the same as regular data packets. That issue is broader than just QoS - applies to firewall filters etc.

In summary:
- o Leverage the existing approach in [RFC2983] for DSCP handling.

13. Congestion Considerations

Additional encapsulation headers does not introduce anything new for Explicit Congestion Notification. It is just like IP-in-IP and IPsec tunnels which is specified in [RFC6040] in terms of how the ECN bits in the inner and outer header are handled when encapsulating and decapsulating packets. Thus new encapsulations can more or less include that by reference.

There are additional considerations around carrying non-congestion controlled traffic. These details have been worked out in [I-D.ietf-mpls-in-udp]. As specified in [RFC5405]: "IP-based traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path Consequently, a tunnel carrying IP-based traffic should already interact appropriately with other traffic sharing the path, and specific congestion control mechanisms for the tunnel are not necessary". Those considerations are being captured in [I-D.ietf-tsvwg-rfc5405bis].

For this reason, where an encapsulation method is used to carry IP traffic that is known to be congestion controlled, the UDP tunnels does not create an additional need for congestion control. Internet IP traffic is generally assumed to be congestion-controlled. Similarly, in general Layer 3 VPNs are carrying IP traffic that is similarly assumed to be congestion controlled.

However, some of the encapsulations (at least NVO3) will be able to carry arbitrary Layer 2 packets to provide an L2 service, in which case one can not assume that the traffic is congestion controlled.

One could handle this by adding some congestion control support to the encapsulation header (one instance of which would end up looking like DCCP). However, if the underlay is well-provisioned and managed as opposed to being arbitrary Internet path, it might be sufficient to have a slower reaction to congestion induced by that traffic. There is work underway on a notion of "circuit breakers" for this purpose. See See [I-D.ietf-tsvwg-circuit-breaker]. Encapsulations which carry arbitrary Layer 2 packets want to consider that ongoing work.

If the underlay is provisioned in such a way that it can guarantee sufficient capacity for non-congestion controlled Layer 2 traffic, then such circuit breakers might not be needed.

Two other considerations appear in the context of these encapsulations as applied to overlay networks:

- o Protect against malicious end stations
- o Ensure fairness and/or measure resource usage across multiple tenants

Those issues are really orthogonal to the encapsulation, in that they are present even when no new encapsulation header is in use. However, the application of the new encapsulations are likely to be in environments where those issues are becoming more important. Hence it makes sense to consider them.

One could make the encapsulation header be extensible to that it can carry sufficient information to be able to measure resource usage, delays, and congestion. The suggestions in the OAM section about a single bit for counter synchronization, and optional timestamps and/or sequence numbers, could be part of such an approach. There might also be additional congestion-control extensions to be carried in the encapsulation. Overall this results in a consideration to be able to have sufficient extensibility in the encapsulation to be handle to handle potential future developments in this space.

Coarse measurements are likely to suffice, at least for circuit-breaker-like purposes, see [I-D.wei-tsvwg-tunnel-congestion-feedback] and [I-D.briscoe-conex-data-centre] for examples on active work in this area via use of ECN. [RFC6040] Appendix C is also relevant. The outer ECN bits seem sufficient (at least when everything uses ECN) to do this course measurements. Needs some more study for the case when there are also drops; might need to exchange counters between ingress and egress to handle drops.

Circuit breakers are not sufficient to make a network with different congestion control when the goal is to provide a predictable service to different tenants. The fallback would be to rate limit different traffic.

In summary:
- o Leverage the existing approach in [RFC6040] for ECN handling.
- o If the encapsulation can carry non-IP, hence non-congestion controlled traffic, then leverage the approach in [I-D.ietf-mpls-in-udp].
- o "Watch this space" for circuit breakers.

14. Header Protection

Many UDP based encapsulations such as VXLAN [RFC7348] either discourage or explicitly disallow the use of UDP checksums. The reason is that the UDP checksum covers the entire payload of the packet and switching ASICs are typically optimized to look at only a small set of headers as the packet passes through the switch. In these case, computing a checksum over the packet is very expensive. (Software endpoints and the NICs used with them generally do not have the same issue as they need to look at the entire packet anyways.)

The lack a header checksum creates the possibility that bit errors can be introduced into any information carried by the new headers. Specifically, in the case of IPv6, the assumption is that a transport layer checksum - UDP in this case - will protect the IP addresses through the inclusion of a pseudoheader in the calculation. This is

different from IPv4 on which many of these encapsulation protocols are initially deployed which contains its own header checksum. In addition to IP addresses, the encapsulation header often contains its own information which is used for addressing packets or other high value network functions. Without a checksum, this information is potentially vulnerable - an issue regardless of whether the packet is carried over IPv4 or IPv6.

Several protocols cite [RFC6935] and [RFC6936] as an exemption to the IPv6 checksum requirements. However, these are intended to be tailored to a fairly narrow set of circumstances - primarily relying on sparseness of the address space to detect invalid values and well managed networks - and are not a one size fits all solution. In these cases, an analysis should be performed of the intended environment, including the probability of errors being introduced and the use of ECC memory in routing equipment.

Conceptually, the ideal solution to this problem is a checksum that covers only the newly added headers of interest. There is little value in the portion of the UDP checksum that covers the encapsulated packet because that would generally be protected by other checksums and this is the expensive portion to compute. In fact, this solution already exists in the form of UDP-Lite and UDP based encapsulations could be easily ported to run on top of it. Unfortunately, the main value in using UDP as part of the encapsulation header is that it is recognized by already deployed equipment for the purposes of ECMP, RSS, and middlebox operations. As UDP-Lite uses a different protocol number than UDP and it is not widely implemented in middleboxes, this value is lost. A possible solution is to incorporate the same partial-checksum concept as UDP-Lite or other header checksum protection into the encapsulation header and continue using UDP as the outer protocol. One potential challenge with this approach is the use of NAT or other form of translation on the outer header will result in an invalid checksum as the translator will not know to update the encapsulation header.

The method chosen to protect headers is often related to the security needs of the encapsulation mechanism. On one hand, the impact of a poorly protected header is not limited to only data corruption but can also introduce a security vulnerability in the form of misdirected packets to an unauthorized recipient. Conversely, high security protocols that already include a secure hash over the valuable portion of the header (such as by encrypting the entire IP packet using IPsec, or some secure hash of the encap header) do not require additional checksum protection as the hash provides stronger assurance than a simple checksum.

If the sender has included a checksum, then the receiver should

verify that checksum or, if incapable, drop the packet. The assumption is that configuration and/or control-plane capability exchanges can be used when different receiver have different checksum validation capabilities.

In summary:
- o Encapsulations need extensibility to be able to add checksum/CRC for the encapsulation header itself.
- o When the encapsulation has a checksum/CRC, include the IPv6 pseudo-header in it.
- o The checksum/CRC can potentially be avoided when cryptographic protection is applied to to the encapsulation.

15. Extensibility Considerations

Protocol extensibility is the concept that a networking protocol may be extended to include new use cases or functionality that were not part of the original protocol specification. Extensibility may be used to add security, control, management, or performance features to a protocol. A solution may allow private extensions for customization or experimentation.

Extending a protocol often implies that a protocol header must carry new information. There are two usual methods to accomplish this:
1. Define or redefine the meaning of existing fields in a protocol header.
2. Add new (optional) fields to the protocol header.

It is also possible to create a new protocol version, but this is more associated with defining a protocol than extending it (IPv6 being a successor to IPv4 is an example of protocol versioning).

In some cases it might be more appropriate to define a new inner protocol which can carry the new functionality instead of extending the outer protocol. Examples where this works well is in the IP/transport split, where the earlier architecture had a single NCP protocol which carried both the hop-by-hop semantics which are now in IP, and the end-to-end semantics which are now in TCP. Such a split is effective when different nodes need to act upon the different information. Applying this for general protocol extensibility through nesting is not well understood, and does result in longer header chains. Furthermore, our experience with IPv6 extension headers [RFC2460] in middleboxes indicates that the approach does not help with middlebox traversal.

Many protocol definitions include some number of reserved fields or bits which can be used for future extension. VXLAN is an example of a protocol that includes reserved bits which are subsequently being

allocated for new purposes. Another technique employed is to repurpose existing header fields with new meanings. A classic example of this is the definition of DSCP code point which redefines the ToS field originally specified in IPv4. When a field is redefined, some mechanism may be needed to ensure that all interested parties agree on the meaning of the field. The techniques of defining meaning for reserved bits or redefining existing fields have the advantage that a protocol header can be kept a fixed length. The disadvantage is that the extensibility is limited. For instance, the number reserved bits in a fixed protocol header is limited. For standard protocols the decision to commit to a definition for a field can be wrenching since it is difficult to retract later. Also, it is difficult to predict a priori how many reserved fields or bits to put into a protocol header to satisfy the extensions create over the lifetime of the protocol.

Extending a protocol header with new fields can be done in several ways.

- o TLVs are a very popular method used in such protocols as IP and TCP. Depending on the type field size and structure, TLVs can offer a virtually unlimited range of extensions. A disadvantage of TLVs is that processing them can be verbose, quite complicated, several validations must often be done for each TLV, and there is no deterministic ordering for a list of TLVs. TCP serves as an example of a protocol where TLVs have been successfully used (i.e. required for protocol operation). IP is an example of a protocol that allows TLVs but are rarely used in practice (router fast paths usually that assume no IP options). Note that TCP TLVs are implemented in software as well as (NIC) hardware handling various forms of TCP offload.
- o Extension headers are closely related to TLVs. These also carry type/value information, but instead of being a list of TLVs within a single protocol header, each one is in its own protocol header. IPv6 extension headers and SFC NSH are examples of this technique. Similar to TLVs these offer a wide range of extensibility, but have similarly complex processing. Another difference with TLVs is that each extension header is idempotent. This is beneficial in cases where a protocol implements a push/pop model for header elements like service chaining, but makes it more difficult group correlated information within one protocol header.
- o A particular form of extension headers are the tags used by IEEE 802 protocols. Those are similar to e.g., IPv6 extension headers but with the key difference that each tag is a fixed length header where the length is implicit in the tag value. Thus as long as a receiver can be programmed with a tag value to length map, it can skip those new tags.

- o Flag-fields are a non-TLV like method of extending a protocol header. The basic idea is that the header contains a set of flags, where each set flags corresponds to optional field that is present in the header. GRE is an example of a protocol that employs this mechanism. The fields are present in the header in the order of the flags, and the length of each field is fixed. Flag-fields are simpler to process compared to TLVs, having fewer validations and the order of the optional fields is deterministic. A disadvantage is that range of possible extensions with flag-fields is smaller than TLVs.

The requirements for receiving unknown or unimplemented extensible elements in an encapsulation protocol (flags, TLVs, optional fields) need to be specified. There are two parties to consider, middle boxes and terminal endpoints of encapsulation (at the decapsulator).

A protocol may allow or expect nodes in a path to modify fields in an encapsulation (example use of this is BIER). In this case, the middleboxes should follow the same requirements as nodes terminating the encapsulation. In the case that middle boxes do not modify the encapsulation, we can assume that they may still inspect any fields of the encapsulation. Missing or unknown fields should be accepted per protocol specification, however it is permissible for a site to implement a local policy otherwise (e.g. a firewall may drop packets with unknown options).

For handling unknown options at terminal nodes, there are two possibilities: drop packet or accept while ignoring the unknown options. Many Internet protocols specify that reserved flags must be set to zero on transmission and ignored on reception. L2TP is example data protocol that has such flags. GRE is a notable exception to this rule, reserved flag bits 1-5 cannot be ignored [RFC2890]. For TCP and IPv4, implementations must ignore optional TLVs with unknown type; however in IPv6 if a packet contains an unknown extension header (unrecognized next header type) the packet must be dropped with an ICMP error message returned. The IPv6 options themselves (encoded inside the destinations options or hop-by-hop options extension header) have more flexibility. There bits in the option code are used to instruct the receiver whether to ignore, silently drop, or drop and send error if the option is unknown. Some protocols define a "mandatory bit" that can is set with TLVs to indicate that an option must not be ignored. Conceptually, optional data elements can only be ignored if they are idempotent and do not alter how the rest of the packet is parsed or processed.

Depending on what type of protocol evolution one can predict, it might make sense to have an way for a sender to express that the

packet should be dropped by a terminal node which does not understand the new information. In other cases it would make sense to have the receiver silently ignore the new info. The former can be expressed by having a version field in the encapsulation, or a notion of "mandatory bit" as discussed above.

A security mechanism which use some form secure hash over the encapsulation header would need to be able to know which extensions can be changed in flight.

In summary:

- o Encapsulations need the ability to be extended to handle e.g., the OAM or security aspects discussed in this document.
- o Practical experience seems to tell us that extensibility mechanisms which are not in use on day one might result in immediate ossification by lack of implementation support. In some cases that has occurred in routers and in other cases in middleboxes. Hence devising ways where the extensibility mechanisms are in use seems important.

16. Layering Considerations

One can envision that SFC might use NVO3 as a delivery/transport mechanism. With more imagination that in turn might be delivered using BIER. Thus it is useful to think about what things look like when we have BIER+NVO3+SFC+payload. Also, if NVO3 is widely deployed there might be cases of NVO3 nesting where a customer uses NVO3 to provide network virtualization e.g., across departments. That customer uses a service provider which happens to use NVO3 to provide transport for their customers.Thus NVO3 in NVO3 might happen.

A key question we set out to answer is what the packets might look like in such a case, and in particular whether we would end up with multiple UDP headers for entropy.

Based on the discussion in the Entropy section, the entropy is associated with the outer delivery IP header. Thus if there are multiple IP headers there would be a UDP header for each one of the IP headers. But SFC does not require its own IP header. So a case of NVO3+SFC would be IP+UDP+NVO3+SFC. A nested NVO3 encapsulation would have independent IP+UDP headers.

The layering also has some implications for middleboxes.

- o A device on the path between the ingress and egress is allowed to transparently inspect all layers of the protocol stack and drop or forward, but not transparently modify anything but the layer in which they operate. What this means is that an IP router is

allowed modify the outer IP ttl and ECN bits, but not the encapsulation header or inner headers and payload. And a BIER router is allowed to modify the BIER header.

- o Alternatively such a device can become visible at a higher layer. E.g., a middlebox could become an decapsulate + function + encapsulate which means it will generate a new encapsulation header.

The design team asked itself some additional questions:

- o Would it make sense to have a common encapsulation base header (for OAM, security?, etc) and then followed by the specific information for NVO3, SFC, BIER? Given that there are separate proposals and the set of information needing to be carried differs, and the extensibility needs might be different, it would be difficult and not that useful to have a common base header.
- o With a base header in place, one could view the different functions (NVO3, SFC, and BIER) as different extensions to that base header resulting in encodings which are more space optimal by not repeating the same base header. The base header would only be repeated when there is an additional IP (and hence UDP) header. That could mean a single length field (to skip to get to the payload after all the encapsulation headers). That might be technically feasible, but it would create a lot of dependencies between different WGs making it harder to make progress. Compare with the potential savings in packet size.

17. Service model

The IP service is lossy and subject to reordering. In order to avoid a performance impact on transports like TCP the handling of packets is designed to avoid reordering packets that are in the same transport flow (which is typically identified by the 5-tuple). But across such flows the receiver can see different ordering for a given sender. That is the case for a unicast vs. a multicast flow from the same sender.

There is a general tussle between the desire for high capacity utilization across a multipath network and the import on packet ordering within the same flow (which results in lower transport protocol performance). That isn’t affected by the introduction of an encapsulation. However, the encapsulation comes with some entropy, and there might be cases where folks want to change that in response to overload or failures. For instance, might want to change UDP source port to try different ECMP route. Such changes can result in packet reordering within a flow, hence would need to be done infrequently and with care e.g., by identifying packet trains.

There might be some applications/services which are not able to handle reordering across flows. The IETF has defined pseudo-wires [RFC3985] which provides the ability to ensure ordering (implemented using sequence numbers and/or timestamps).

Architectural such services would make sense, but as a separate layer on top of an encapsulation protocol. They could be deployed between ingress and egress of a tunnel which uses some encaps. Potentially the tunnel control points at the ingress and egress could become a platform for fixing suboptimal behavior elsewhere in the network. That would clearly be undesirable in the general case. However, handling encapsulation of non-IP traffic hence non-congestion-controlled traffic is likely to be required, which implies some fairness and/or QoS policing on the ingress and egress devices.

But the tunnels could potentially do more like increase reliability (retransmissions, FEC) or load spreading using e.g. MP-TCP between ingress and egress.

18. Hardware Friendly

Hosts, switches and routers often leverage capabilities in the hardware to accelerate packet encapsulation, decapsulation and forwarding.

Some design considerations in encapsulation that leverage these hardware capabilities may result in more efficiently packet processing and higher overall protocol throughput.

While "hardware friendliness" can be viewed as unnecessary considerations for a design, part of the motivation for considering this is ease of deployment; being able to leverage existing NIC and switch chips for at least a useful subset of the functionality that the new encapsulation provides. The other part is the ease of implementing new NICs and switch/router chips that support the encapsulation at ever increasing line rates.

[disclaimer] There are many different types of hardware in any given network, each maybe better at some tasks while worse at others. We would still recommend protocol designers to examine the specific hardware that are likely to be used in their networks and make decisions on a case by case basis.

Some considerations are:

- o Keep the encap header small. Switches and routers usually only read the first small number of bytes into the fast memory for quick processing and easy manipulation. The bulk of the packets

are usually stored in slow memory. A big encap header may not fit and additional read from the slow memory will hurt the overall performance and throughput.

- o Put important information at the beginning of the encapsulation header. The reasoning is similar as explained in the previous point. If important information are located at the beginning of the encapsulation header, the packet may be processed with smaller number of bytes to be read into the fast memory and improve performance.
- o Avoid full packet checksums in the encapsulation if possible. Encapsulations should instead consider adding their own checksum which covers the encapsulation header and any IPv6 pseudo-header. The motivation is that most of the switch/router hardware make switching/forwarding decisions by reading and examining only the first certain number of bytes in the packet. Most of the body of the packet do not need to be processed normally. If we are concerned of preventing packet to be misdelivered due to memory errors, consider only perform header checksums. Note that NIC chips can typically already do full packet checksums for TCP/UDP, while adding a header checksum might require adding some hardware support.
- o Place important information at fixed offset in the encapsulation header. Packet processing hardware may be capable of parallel processing. If important information can be found at fixed offset, different part of the encapsulation header may be processed by different hardware units in parallel (for example multiple table lookups may be launched in parallel). It is easier for hardware to handle optional information when the information, if present, can be found in ideally one place, but in general, in as few places as possible. That facilitates parallel processing. TLV encoding with unconstrained order typically does not have that property.
- o Limit the number of header combinations. In many cases the hardware can explore different combinations of headers in parallel, however there is some added cost for this.

18.1. Considerations for NIC offload

This section provides guidelines to provide support of common offloads for encapsulation in Network Interface Cards (NICs). Offload mechanisms are techniques that are implemented separately from the normal protocol implementation of a host networking stack and are intended to optimize or speed up protocol processing. Hardware offload is performed within a NIC device on behalf of a host.

There are three basic offload techniques of interest:

- o Receive multi queue
- o Checksum offload
- o Segmentation offload

18.1.1. Receive multi-queue

Contemporary NICs support multiple receive descriptor queues (multi-queue). Multi-queue enables load balancing of network processing for a NIC across multiple CPUs. On packet reception, a NIC must select the appropriate queue for host processing. Receive Side Scaling (RSS) is a common method which uses the flow hash for a packet to index an indirection table where each entry stores a queue number.

UDP encapsulation, where the source port is used for entropy, should be compatible with multi-queue NICs that support five-tuple hash calculation for UDP/IP packets as input to RSS. The source port ensures classification of the encapsulated flow even in the case that the outer source and destination addresses are the same for all flows (e.g. all flows are going over a single tunnel).

18.1.2. Checksum offload

Many NICs provide capabilities to calculate standard ones complement payload checksum for packets in transmit or receive. When using encapsulation over UDP there are at least two checksums that may be of interest: the encapsulated packet’s transport checksum, and the UDP checksum in the outer header.

18.1.2.1. Transmit checksum offload

NICs may provide a protocol agnostic method to offload transmit checksum (NETIF_F_HW_CSUM in Linux parlance) that can be used with UDP encapsulation. In this method the host provides checksum related parameters in a transmit descriptor for a packet. These parameters include the starting offset of data to checksum, the length of data to checksum, and the offset in the packet where the computed checksum is to be written. The host initializes the checksum field to pseudo header checksum. In the case of encapsulated packet, the checksum for an encapsulated transport layer packet, a TCP packet for instance, can be offloaded by setting the appropriate checksum parameters.

NICs typically can offload only one transmit checksum per packet, so simultaneously offloading both an inner transport packet’s checksum and the outer UDP checksum is likely not possible. In this case setting UDP checksum to zero (per above discussion) and offloading the inner transport packet checksum might be acceptable.

There is a proposal in [I-D.herbert-remotecsumoffload] to leverage NIC checksum offload when an encapsulator is co-resident with a host.

18.1.2.2. Receive checksum offload

Protocol encapsulation is compatible with NICs that perform a protocol agnostic receive checksum (CHECKSUM_COMPLETE in Linux parlance). In this technique, a NIC computes a ones complement checksum over all (or some predefined portion) of a packet. The computed value is provided to the host stack in the packet's receive descriptor. The host driver can use this checksum to "patch up" and validate any inner packet transport checksum, as well as the outer UDP checksum if it is non-zero.

Many legacy NICs don't provide checksum-complete but instead provide an indication that a checksum has been verified (CHECKSUM_UNNECESSARY in Linux). Usually, such validation is only done for simple TCP/IP or UDP/IP packets. If a NIC indicates that a UDP checksum is valid, the checksum-complete value for the UDP packet is the "not" of the pseudo header checksum. In this way, checksum-unnecessary can be converted to checksum-complete. So if the NIC provides checksum-unnecessary for the outer UDP header in an encapsulation, checksum conversion can be done so that the checksum-complete value is derived and can be used by the stack to validate an checksums in the encapsulated packet.

18.1.3. Segmentation offload

Segmentation offload refers to techniques that attempt to reduce CPU utilization on hosts by having the transport layers of the stack operate on large packets. In transmit segmentation offload, a transport layer creates large packets greater than MTU size (Maximum Transmission Unit). It is only at much lower point in the stack, or possibly the NIC, that these large packets are broken up into MTU sized packet for transmission on the wire. Similarly, in receive segmentation offload, small packets are coalesced into large, greater than MTU size packets at a point low in the stack receive path or possibly in a device. The effect of segmentation offload is that the number of packets that need to be processed in various layers of the stack is reduced, and hence CPU utilization is reduced.

18.1.3.1. Transmit Segmentation Offload

Transmit Segmentation Offload (TSO) is a NIC feature where a host provides a large (larger than MTU size) TCP packet to the NIC, which in turn splits the packet into separate segments and transmits each one. This is useful to reduce CPU load on the host.

The process of TSO can be generalized as:

- o Split the TCP payload into segments which allow packets with size less than or equal to MTU.
- o For each created segment:
  1. Replicate the TCP header and all preceding headers of the original packet.
  2. Set payload length fields in any headers to reflect the length of the segment.
  3. Set TCP sequence number to correctly reflect the offset of the TCP data in the stream.
  4. Recompute and set any checksums that either cover the payload of the packet or cover header which was changed by setting a payload length.

Following this general process, TSO can be extended to support TCP encapsulation UDP. For each segment the Ethernet, outer IP, UDP header, encapsulation header, inner IP header if tunneling, and TCP headers are replicated. Any packet length header fields need to be set properly (including the length in the outer UDP header), and checksums need to be set correctly (including the outer UDP checksum if being used).

To facilitate TSO with encapsulation it is recommended that optional fields should not contain values that must be updated on a per segment basis-- for example an encapsulation header should not include checksums, lengths, or sequence numbers that refer to the payload. If the encapsulation header does not contain such fields then the TSO engine only needs to copy the bits in the encapsulation header when creating each segment and does not need to parse the encapsulation header.

18.1.3.2. Large Receive Offload

Large Receive Offload (LRO) is a NIC feature where packets of a TCP connection are reassembled, or coalesced, in the NIC and delivered to the host as one large packet. This feature can reduce CPU utilization in the host.

LRO requires significant protocol awareness to be implemented correctly and is difficult to generalize. Packets in the same flow need to be unambiguously identified. In the presence of tunnels or network virtualization, this may require more than a five-tuple match (for instance packets for flows in two different virtual networks may have identical five-tuples). Additionally, a NIC needs to perform validation over packets that are being coalesced, and needs to fabricate a single meaningful header from all the coalesced packets.

The conservative approach to supporting LRO for encapsulation would

be to assign packets to the same flow only if they have identical five-tuple and were encapsulated the same way. That is the outer IP addresses, the outer UDP ports, encapsulated protocol, encapsulation headers, and inner five tuple are all identical.

18.1.3.3. In summary:

In summary, for NIC offload:

- o The considerations for using full UDP checksums are different for NIC offload than for implementations in forwarding devices like routers and switches.
- o Be judicious about encapsulations that change fields on a per-packet basis, since such behavior might make it hard to use TSO.

19. Middlebox Considerations

This document has touched upon middleboxes in different section. The reason for this is as encapsulations get widely deployed one would expect different forms of middleboxes might become aware of the encapsulation protocol just as middleboxes have been made aware of other protocols where there are business and deployment opportunities. Such middleboxes are likely to do more than just drop packets based on the UDP port number used by an encapsulation protocol.

We note that various forms of encapsulation gateways that stitch one encapsulation protocol together with another form of protocol could have similar effects.

An example of a middlebox that could see some use would be an NVO3-aware firewall that would filter on the VNI IDs to provide some defense in depth inside or across NVO3 datacenters.

A question for the IETF is whether we should document what to do or what not to do in such middleboxes. This document touches on areas of OAM and ECMP as it relates to middleboxes and it might make sense to document how encapsulation-aware middleboxes should avoid unintended consequences in those (and perhaps other) areas.

In summary:

- o We are likely to see middleboxes that at least parse the headers for succesful new encapsulations.
- o Should the IETF document considerations for what not to do in such middleboxes?

20. Related Work

The IETF and industry has defined encapsulations for a long time, with examples like GRE [RFC2890], VXLAN [RFC7348], and NVGRE [I-D.sridharan-virtualization-nvgre] being able to carry arbitrary Ethernet payloads, and various forms of IP-in-IP and IPsec encapsulations that can carry IP packets. As part of NVO3 there has been additional proposals like Geneve [I-D.gross-geneve] and GUE [I-D.herbert-gue] which look at more extensibility. NSH [I-D.quinn-sfc-nsh] is an example of an encapsulation that tries to provide extensibility mechanisms which target both hardware and software implementations.

There is also a large body of work around MPLS encapsulations [RFC3032]. The MPLS-in-UDP work [I-D.ietf-mpls-in-udp] and GRE over UDP [I-D.ietf-tsvwg-gre-in-udp-encap] have worked on some of the common issues around checksum and congestion control. MPLS also introduced a entropy label [RFC6790]. There is also a proposal for MPLS encryption [I-D.farrelll-mpls-opportunistic-encrypt].

The idea to use a UDP encapsulation with a UDP source port for entropy for the underlay routers' ECMP dates back to LISP [RFC6830].

The pseudo-wire work [RFC3985] is interesting in the notion of layering additional services/characteristics such as ordered delivery or timely deliver on top of an encapsulation. That layering approach might be useful for the new encapsulations as well. For instance, the control word [RFC4385]. There is also material on congestion control for pseudo-wires in [I-D.ietf-pwe3-congcons].

Both MPLS and L2TP [RFC3931] rely on some control or signaling to establish state (for the path/labels in the case of MPLS, and for the session in the case of L2TP). The NVO3, SFC, and BIER encapsulations will also have some separation between the data plane and control plane, but the type of separation appears to be different.

IEEE 802.1 has defined encapsulations for L2 over L2, in the form of Provider backbone Bridging (PBB) [IEEE802.1Q-2014] and Equal Cost Multipath (ECMP) [IEEE802.1Q-2014]. The latter includes something very similar to the way the UDP source port is used as entropy: "The flow hash, carried in an F-TAG, serves to distinguish frames belonging to different flows and can be used in the forwarding process to distribute frames over equal cost paths"

TRILL, which is also a L2 over L2 encapsulation, took a different approach to entropy but preserved the ability for OAM frames [RFC7174] to use the same entropy hence ECMP path as data frames. In [I-D.ietf-trill-oam-fm] there 96 bytes of headers for entropy in the

OAM frames, followed by the actual OAM content. This ensures that any headers, which fit in those 96 bytes except the OAM bit in the TRILL header, can be used for ECMP hashing.

As encapsulations evolve there might be a desire to fit multiple inner packets into one outer packet. The work in [I-D.saldana-tsvwg-simplemux] might be interesting for that purpose.

21. Acknowledgements

The authors acknowledge the comments from Alia Atlas, Fred Baker, David Black, Bob Briscoe, Stewart Bryant, Mike Cox, Andy Malis, Radia Perlman, Michael Smith, and Lucy Yong.

22. Open Issues

- o Middleboxes:
  - * Due to OAM there are constraints on middleboxes in general. If middleboxes inspect the packet past the outer IP+UDP and encapsulation header and look for inner IP and TCP/UDP headers, that might violate the assumption that OAM packets will be handled the same as regular data packets. That issue is broader than just QoS - applies to firewall filters etc.
  - * Firewalls looking at inner payload? How does that work for OAM frames? Even if it only drops ... TRILL approach might be an option? Would that encourage more middleboxes making the network more fragile?
  - * Editorially perhaps we should pull the above two into a separate section about middlebox considerations?
- o Next-protocol indication - should it be common across different encapsulation headers? We will have different ways to indicate the presence of the first encapsulation header in a packet (could be a UDP destination port, an Ethernet type, etc depending on the outer delivery header). But for the next protocol past an encapsulation header one could envision creating or adoption a common scheme. Such a would also need to be able to identify following headers like Ethernet, IPv4/IPv6, ESP, etc.
- o Common OAM error reporting protocol?
- o There is discussion about timestamps, sequence numbers, etc in three different parts of the document. OAM, Congestion Considerations, and Service Model, where the latter argues that a pseudo-wire service should really be layered on top of the encapsulation using its own header. Those recommendations seem to be at odds with each other. Do we envision sequence numbers, timestamps, etc as potential extensions for OAM and CC? If so, those extensions could be used to provide a service which doesn’t

reorder packets.

23. Change Log

The changes from draft-rtg-dt-encap-01 based on feedback at the Dallas IETF meeting:

- o Setting the context that not all common issues might apply to all encapsulations, but that they should all be understood before being dismissed.
- o Clarified that IPv6 flow label is useful for entropy in combination with a UDP source port.
- o Editorially added a "summary" set of bullets to most sections.
- o Editorial clarifications in the next protocol section to more clearly state the three areas.
- o Folded the two next protocol sections into one.
- o Mention the MPLS first nibble issue in the next protocol section.
- o Mention that viewing the next protocol as an index to a table with processing instructions can provide additional flexibility in the protocol evolution.
- o For the OAM "don’t forward to end stations" added that defining a bit seems better than using a special next-protocol value.
- o Added mention of DTLS in addition to IPsec for security.
- o Added some mention of IPv6 hob-by-hop options of other headers than potentially can be copied from inner to outer header.
- o Added text on architectural considerations when it might make sense to define an additional header/protocol as opposed to using the extensibility mechanism in the existing encapsulation protocol.
- o Clarified the "unconstrained TLVs" in the hardware friendly section.
- o Clarified the text around checksum verification and full vs. header checksums.
- o Added wording that the considerations might apply for encaps outside of the routing area.
- o Added references to draft-ietf-pwe3-congcons, draft-ietf-tsvwg-rfc5405bis, RFC2473, and RFC7325
- o Removed reference to RFC3948.
- o Updated the acknowledgements section.
- o Added this change log section.

24. References

24.1. Normative References

[I-D.ietf-tsvwg-rfc5405bis]
Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage

Guidelines", draft-ietf-tsvwg-rfc5405bis-02 (work in progress), April 2015.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, September 2000.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC3931] Lau, J., Townsley, M., and I. Goyret, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, March 2005.

[RFC3985] Bryant, S. and P. Pate, "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, February 2006.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, April 2013.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement

for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.

[RFC7174] Salam, S., Senevirathne, T., Aldrin, S., and D. Eastlake, "Transparent Interconnection of Lots of Links (TRILL) Operations, Administration, and Maintenance (OAM) Framework", RFC 7174, May 2014.

[RFC7325] Villamizar, C., Kompella, K., Amante, S., Malis, A., and C. Pignataro, "MPLS Forwarding Compliance and Performance Requirements", RFC 7325, August 2014.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.

[RFC7364] Narten, T., Gray, E., Black, D., Fang, L., Kreeger, L., and M. Napierala, "Problem Statement: Overlays for Network Virtualization", RFC 7364, October 2014.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, October 2014.

24.2. Informative References

[I-D.briscoe-conex-data-centre]
Briscoe, B. and M. Sridharan, "Network Performance Isolation in Data Centres using Congestion Policing", draft-briscoe-conex-data-centre-02 (work in progress), February 2014.

[I-D.farrelll-mpls-opportunistic-encrypt]
Farrel, A. and S. Farrell, "Opportunistic Security in MPLS Networks", draft-farrelll-mpls-opportunistic-encrypt-04 (work in progress), January 2015.

[I-D.gross-geneve]
Gross, J., Sridhar, T., Garg, P., Wright, C., Ganga, I., Agarwal, P., Duda, K., Dutt, D., and J. Hudson, "Geneve: Generic Network Virtualization Encapsulation", draft-gross-geneve-02 (work in progress), October 2014.

[I-D.herbert-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-herbert-gue-03 (work in progress),

March 2015.

[I-D.herbert-remotecsumoffload]
Herbert, T., "Remote checksum offload for encapsulation", draft-herbert-remotecsumoffload-01 (work in progress), November 2014.

[I-D.ietf-mpls-in-udp]
Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", draft-ietf-mpls-in-udp-11 (work in progress), January 2015.

[I-D.ietf-nvo3-arch]
Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Overlay Networks (NVO3)", draft-ietf-nvo3-arch-03 (work in progress), March 2015.

[I-D.ietf-pwe3-congcons]
Stein, Y., Black, D., and B. Briscoe, "Pseudowire Congestion Considerations", draft-ietf-pwe3-congcons-02 (work in progress), July 2014.

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-08 (work in progress), May 2015.

[I-D.ietf-sfc-problem-statement]
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-13 (work in progress), February 2015.

[I-D.ietf-trill-oam-fm]
Senevirathne, T., Finn, N., Salam, S., Kumar, D., Eastlake, D., Aldrin, S., and L. Yizhou, "TRILL Fault Management", draft-ietf-trill-oam-fm-11 (work in progress), October 2014.

[I-D.ietf-tsvwg-circuit-breaker]
Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-01 (work in progress), March 2015.

[I-D.ietf-tsvwg-gre-in-udp-encap]
Crabbe, E., Yong, L., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", draft-ietf-tsvwg-gre-in-udp-encap-06 (work in progress), March 2015.

[I-D.ietf-tsvwg-port-use]
Touch, J., "Recommendations on Using Assigned Transport Port Numbers", draft-ietf-tsvwg-port-use-11 (work in progress), April 2015.

[I-D.quinn-sfc-nsh]
Quinn, P., Guichard, J., Surendra, S., Smith, M., Henderickx, W., Nadeau, T., Agarwal, P., Manur, R., Chauhan, A., Halpern, J., Majee, S., Elzur, U., Melman, D., Garg, P., McConnell, B., Wright, C., and K. Kevin, "Network Service Header", draft-quinn-sfc-nsh-07 (work in progress), February 2015.

[I-D.saldana-tsvwg-simplemux]
Saldana, J., "Simplemux. A generic multiplexing protocol", draft-saldana-tsvwg-simplemux-02 (work in progress), January 2015.

[I-D.shepherd-bier-problem-statement]
Shepherd, G., Dolganow, A., and a. arkadiy.gulko@thomsonreuters.com, "Bit Indexed Explicit Replication (BIER) Problem Statement", draft-shepherd-bier-problem-statement-02 (work in progress), February 2015.

[I-D.sridharan-virtualization-nvgre]
Garg, P. and Y. Wang, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-08 (work in progress), April 2015.

[I-D.wei-tsvwg-tunnel-congestion-feedback]
Wei, X., Zhu, L., and L. Deng, "Tunnel Congestion Feedback", draft-wei-tsvwg-tunnel-congestion-feedback-03 (work in progress), October 2014.

[I-D.wijnands-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-wijnands-bier-architecture-05 (work in progress), March 2015.

[I-D.wijnands-mpls-bier-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-wijnands-mpls-bier-encapsulation-02 (work in progress), December 2014.

[I-D.xu-bier-encapsulation]
Xu, X., Somasundaram, S., Jacquenet, C., and R. Raszuk, "BIER Encapsulation", draft-xu-bier-encapsulation-02 (work in progress), February 2015.

[IEEE802.1Q-2014]
IEEE, "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks", IEEE Std 802.1Q-2014, 2014, <http://www.ieee802.org/1/pages/802.1Q-2014.html>.

(Access Controlled link within page)

Authors' Addresses

Erik Nordmark
Arista Networks
5453 Great America Parkway
Santa Clara, CA 95054
USA

Email: nordmark@arista.com

Albert Tian
Ericsson Inc.
300 Holger Way
San Jose, California 95134
USA

Email: albert.tian@ericsson.com

Jesse Gross
VMware
3401 Hillview Ave.
Palo Alto, CA 94304
USA

Email: jgross@vmware.com

Jon Hudson
Brocade Communications Systems, Inc.
130 Holger Way
San Jose, CA 95134
USA

Email: jon.hudson@gmail.com

Lawrence Kreeger
Cisco Systems, Inc.
170 W. Tasman Avenue
San Jose, CA 95134
USA

Email: kreeger@cisco.com

Pankaj Garg
Microsoft
1 Microsoft Way
Redmond, WA 98052
USA

Email: pankajg@microsoft.com

Patricia Thaler
Broadcom Corporation
3151 Zanker Road
San Jose, CA 95134
USA

Email: pthaler@broadcom.com

Tom Herbert
Google
1600 Amphitheatre Parkway
Mountain View, CA
USA

Email: therbert@google.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

A. Jain
A. Terzis
Google
N. Sprecher
S. Arunachalam
Nokia Networks
K. Smith
G. Klas
Vodafone

March 13, 2017

Requirements and reference architecture for Mobile Throughput Guidance Exposure

draft-sprecher-mobile-tg-exposure-req-arch-03.txt

Abstract

Rapidly-varying conditions in a cellular network can cause problems for the Transmission Control Protocol (TCP), which in turn can degrade application performance.

This document presents the problem statement and proposes solution principles. It specifies the requirements and reference architecture for a mobile throughput guidance exposure mechanism that can be used to assist TCP in cellular networks, ensuring better network efficiency and enhanced service delivery performance.

The proposed mechanism can be applied to any content or TCP/IP based application content delivery. This document describes the applicability of the mechanism for Intelligent Video Acceleration over cellular networks.

Status of This Memo

This Internet-Draft will expire on September 14, 2017.

Copyright Notice

Table of Contents

1. Introduction

Mobile-edge Computing provides IT and cloud-computing capabilities within the Radio Access Network in close proximity to mobile subscribers [MEC_White_Paper]. Mobile Throughput Guidance is an solution building on top of MEC concepts to improve customer experience during mobile Internet sessions, by making explicit what

range of bandwidth the mobile access link is likely to sustain [MTG_GSMA_White_paper].

The following sub-sections present the problem statement and the solution principles for Mobile Throughput Guidance.

1.1. Contributing Authors

The editors gratefully acknowledge the following additional contributors: Hannu Flinck, Helen Parsons, Peter Cosimini and Ram Gopal.

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 RFC 2119 [RFC2119].

1.3. Acronyms and Abbreviations

HTTP Hypertext Transmission Protocol
IP Internet Protocol
LTE Long Term Evolution
RAN Radio Access Network
RTT Round Trip Time
TCP Transmission Control Protocol
UE User Equipment

1.4. Problem statement

Inefficient use of a cellular network’s resources degrades application performance, delivery of content and user experience.

Cellular networks are often required to deliver large, high bandwidth files to end users, e.g from streaming media content providers. If the available throughput from the Radio Access Network (RAN) to the User Equipment (UE) falls below the bandwidth required then files are delivered too slowly, resulting in a bad user experience. It may be possible to take avoiding action and so limit the impact on the network and the user experience. However to be able to do this in an accurate and timely fashion, information on the available throughput is required.

Internet media and file delivery are typically streamed or downloaded today using Hypertext Transmission Protocol (HTTP) over the TCP. The behavior of TCP assumes that network congestion is the primary cause for packet loss and high delay. This may not be the case in cellular networks where the bandwidth available for each UE can vary by an

order of magnitude within a few seconds due to changes in the underlying radio channel conditions. Such changes can be caused by the movement of devices or interference, as well as changes in system load due to bursty traffic sources or when other devices enter and leave the network. On the other hand, packet losses tend to be sporadic and temporary; retransmission mechanisms at the physical and link layers repair most packet corruptions.

1.5. Solution Principles

This document proposes that the cellular network could provide near real-time information on "Throughput Guidance" to the TCP server; this throughput guidance information would indicate the throughput estimated to be available at the radio downlink interface (between the RAN and the UE) for the TCP connection.

While the implementation details will vary according to the cellular access network technology, the resource allocation can be abstracted as the capacity of the "radio link" between the network and the UE. For example, in the case of an LTE network, the number of physical resource blocks allocated to a UE, along with the modulation scheme and coding rate used, can be translated into radio link capacity in Megabits per second (Mbps). It can also include the quality of the "radio link" which is reported by the UE.

The TCP server can use this explicit information to inform several congestion control decisions. For example: (1) selecting the initial window size, (2) deciding the value of the congestion window during the congestion avoidance phase, and (3) reducing the size of the congestion window when the conditions on the "radio link" deteriorate. In other words, with this additional information, TCP does neither have to congest the network when probing for available resource, nor rely on heuristics to reduce its sending rate after a congestion episode.

The same explicit information can also be used to optimize application behavior given the available resources. For example, when video is encoded in multiple bitrates, the application server can select the appropriate encoding for the network conditions.

Note that the throughput estimation for the upstream traffic between the UE and the RAN, and the throughput of the network path between the RAN and the server communicating with the UE are beyond the scope of the document.

It is also important to note that the validity of the throughput guidance and the distance between the originating server and the cellular network (in terms of the number of Internet hops) are

inversely proportional. This is due to the fact that the latency incurred at each hop increases the time that elapses between issuing and consuming the guidance.

2. Requirements

The requirements set out in section 2.1 are for the behavior of the mobile throughput guidance exposure mechanism and the related functional elements. The related security requirements are specified in section 2.2.

2.1. Requirements on the Mobile Throughput Guidance Exposure Mechanism

1. The throughput guidance information SHALL indicate the expected available bandwidth in the downlink interface. Depending on the solution mechanism, the information MAY be provided per TCP flow or per user. If the solution mechanism supports both options, then granularity SHOULD be configurable.

2. The throughput guidance information SHALL be provided for TCP based traffic.

3. A functional element, residing in the RAN and acting as Throughput Guidance Provider, SHOULD supply the TCP server a near real-time indication (in sub-seconds) on the throughput estimated to be available at the radio downlink interface (i.e., mobile throughput guidance information). It SHOULD keep up with the rapid changes in the radio network conditions, the network traffic and the user movement, in order to provide the most accurate guidance information. Throughput Guidance Provider SHALL access to Network Radio Information.

4. The introduction of the Throughput Guidance exposure mechanism SHALL NOT require any update to the TCP client software.

5. The mobile throughput guidance exposure mechanism SHALL work when the user traffic is end-to-end encrypted (e.g., HTTPS, etc.). This requirement is compliant with the IAB Statement on Internet Confidentiality (see [IAB_Statement]), saying that the IAB "strongly encourage developers to include encryption in their implementations, and to make them encrypted by default. We similarly encourage network and service operators to deploy encryption where it is not yet deployed, and we urge firewall policy administrators to permit encrypted traffic.".

6. The mobile throughput guidance exposure mechanism SHALL NOT adversely impact the behavior of the TCP flows (e.g., it SHOULD

NOT cause an increase in retransmissions or degradation in performance, etc.).

7. The mobile throughput guidance exposure mechanism SHALL ensuring end-to-end consistency of the TCP flow control. The throughput guidance information SHALL NOT override the TCP flow window limits (e.g. min(cwnd, rwnd)) or timer values.

8. The throughput guidance information SHALL be opaque to the intermediate elements between the Throughput Guidance Provider and the TCP server. The intermediate elements SHOULD NOT modify or remove the throughput guidance information.

9. The TCP server MAY reside within the mobile operator’s network (behind the mobile core network) or in the Internet.

10. The TCP server MAY use the mobile throughput guidance information to assist TCP.

11. It SHOULD be possible for the TCP server to provide the exposed mobile throughput guidance information to an authorized higher layer application. The application may use the mobile throughput guidance information to optimize its behavior.

12. The Throughput Guidance Provider SHOULD provide the mobile throughput guidance information periodically, starting from the initiation of the flow.

13. The frequency (in milliseconds) at which mobile throughput guidance needs to be exposed SHALL be configurable.

14. The mobile Throughput Guidance Provider SHALL be able to supply mobile throughput guidance information to more than one TCP server simultaneously, with independent configurable parameters for each server.

15. There SHOULD be a mechanism to configure the Mobile Throughput Guidance Provider with a list of TCP flows for which mobile throughput guidance information shall be exposed.

16. The mobile throughput guidance exposure mechanism SHOULD ensure backward compatibility. Normal TCP processing at the TCP server SHOULD be performed if the TCP server does not recognize the throughput guidance information.

17. The mobile throughput guidance exposure mechanism MUST be extensible, ensuring that additional information can be provided in the future in a non-disruptive, backward-compatible way.

2.2. Security requirements

1. A trustful relationship between the Mobile Throughput Provider and the TCP server SHOULD be formed before any information is exposed.

2. There SHOULD be a mechanism to configure the Mobile Throughput Guidance Provider with a list of destinations to which throughput guidance should be provided.

3. The identity of the Mobile Throughput Guidance Provider SHALL be explicitly known to the TCP server which receives the information. The TCP server SHALL be able to authenticate the identity of the Mobile Throughput Guidance Provider. The Mobile Throughput Guidance Provider MUST NOT reveal any other identity or address of network elements that can compromise the security of the network.

4. The mobile throughput guidance information SHOULD be secured to ensure confidentiality and integrity.

5. There SHOULD be a mechanism to configure the required security level and parameters for the encryption and the authentication if supported.

6. The exposure of the Mobile throughput guidance information SHALL NOT introduce any additional security threats and privacy concerns to the mobile operator’s network, the Internet and the users.

7. The throughput guidance SHOULD be treated only as an estimate to the optimization algorithm running at the TCP server. The TCP server that receives this information SHOULD NOT assume that it is always accurate and up to date. Specifically, the TCP server SHOULD check the validity of the information received and if it finds it erroneous it SHOULD discard it and possibly take other corrective actions (e.g., discard all future throughput guidance information from a particular IP prefix).

3. Reference Architecture

Figure 1 below, depicts the functional elements and their interfaces that comprise the mobile network guidance solution (based on the requirements for mobile throughput guidance).

A Throughput Guidance Provider functional element signals to the TCP server the information on the (near-real time) throughput estimated

to be available at the radio downlink interface. The TCP server resides within the mobile operator’s network or in the Internet.

Note that the Throughput Guidance Provider functional element and the TCP server can belong to the same Administrative System (AS) or to different Administrative Systems.

The TCP server MAY use the information to optimize the TCP behavior. The information MAY also be used by the application to adapt its behavior accordingly and to optimize service delivery performance.

```
                           TCP flow behaviour based on
  +-+-+-+-+-+-+-+        Throughput Guidance Information     +-+-+-+-+-+-+-+
  |             |  <---------------------------------------- |             |
  |  TCP client |        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |  TCP Server |
  |             |        | Througput Guidance  |     (x)     |             |
  |             |        |     Provider        | ----------> |             |
  +-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+

  UE <---------------  RAN  -------------------->       x = Mobile Thourghput
                                                             Guidance Signaling
```

Mobile Throughput Guidance Reference Architecture

Figure 1

The information source and the algorithm used by the Throughput Guidance Provider to calculate the throughput guidance are beyond the scope of this document.

The TCP server MAY use the throughput guidance information to assist TCP in any of the following ways:

- o Determine the size of the initial congestion window
- o Determine when to exit the slow start phase
- o Determine the size of the congestion window during the congestion avoidance phase
- o Determine the size of the window after a congestion event

4. Applicability to Mobile Video Delivery Optimization

The mobile throughput guidance exposure mechanism applies to mobile video delivery optimization.

In this use case the Throughput Guidance Provider sends to the video server throughput guidance information for a TCP flow. The video server may use this information to assist TCP congestion control decisions, for example in selecting the initial congestion window size, and adjusting the size of the congestion window when the conditions on the radio link change. In other words, with this additional information, TCP does not need to overload the network when probing for available resources, nor does it need to rely on heuristics to reduce its sending rate after a congestion episode. Slow start and buffering of content delivery can be eliminated.

The same information may also be used to ensure that the application level coding matches the estimated capacity at the radio downlink.

The aim of all of these improvements is to enhance the end user’s quality of experience. For example, the content’s time-to-start as well as video buffering occurrences can be reduced, the utilization of the radio network’s resources and its throughput can be optimized, etc.

5. Manageability considerations

Manageability of mobile throughput guidance exposure will be discussed in the solution documents. Section 2 specifies a set of requirements on the management of the mobile throughput guidance exposure functional elements and protocol operation.

6. Security considerations

The exposure of mobile throughput guidance information from the cellular network to the TCP server introduces a set of security considerations.

As per requirement #3 in section 2.2, the TCP server SHALL be able to authenticate the identity of the Mobile Throughput Guidance Provider. The Mobile Throughput Guidance Provider MUST NOT reveal any other identity or address of network elements that can compromise the security of the network.

Furthermore, the throughput guidance information should be treated only as an estimate to the congestion control algorithm running at the transport endpoint. The endpoint that receives this information should not assume that it is always correct and accurate.

Specifically, endpoints should check the authenticity and integrity of the information received and if they find it erroneous they should discard it and possibly take other corrective actions (e.g., discard all future throughput guidance information from a particular IP prefix).

One way to check if the throughput guidance information overestimates the capacity available on the radio link is to check whether any packet losses or other signs of congestion (e.g., increasing RTT) occur after the guidance is used. Notably, the same mechanism can be used to deal with bottlenecks in other parts of the end-to-end network path. To check if the throughput guidance underestimates the available network capacity, the source can periodically attempt to send faster and then check for signs of congestion.

Section 2 above, specifies a set of requirements on the mobile throughput guidance exposure protocol to ensure secured communication and operation.

7. IANA considerations

This requirements and architecture document does not introduce any requests for IANA actions.

8. Acknowledgements

We would like to thank Peter Szilagyi, Meir Cohen and Csaba Vulkan for conversations on these issues.

9. References

9.1. Normative References

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

[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>.

[RFC6994] Touch, J., "Shared Use of Experimental TCP Options", RFC 6994, DOI 10.17487/RFC6994, August 2013, <http://www.rfc-editor.org/info/rfc6994>.

9.2. Informative References

[I-D.narten-iana-considerations-rfc2434bis]
Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", draft-narten-iana-considerations-rfc2434bis-09 (work in progress), March 2008.

[IAB_Statement]
"IAB Statement on Internet Confidentiality", November 2014, <http://www.ietf.org/mail-archive/web/ietf-announce/current/msg13460.html>.

[MEC_White_Paper]
ETSI, "Mobile-Edge Computing - Introductory Technical White Paper", September 2014, <http://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf>.

[MTG_GSMA_White_paper]
GSMA, "Mobile Throughput Guidance White Paper", March 2017, <https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

[RFC4413] West, M. and S. McCann, "TCP/IP Field Behavior", RFC 4413, DOI 10.17487/RFC4413, March 2006, <http://www.rfc-editor.org/info/rfc4413>.

Authors' Addresses

Ankur Jain
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Phone: +1-925-526-5879
Email: jankur@google.com

Andreas Terzis
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Phone: +1-650-214-5270
Email: aterzis@google.com

Nurit Sprecher
Nokia Networks
Hod HaSharon
IL

Phone: +97297751229
Email: nurit.sprecher@nsn.com

Swaminathan Arunachalam
Nokia Networks
Irving
FUS

Phone: +19723303204
Email: swaminathan.arunachalam@nsn.com

Kevin Smith
Vodafone
One Kingdom Street, Paddington Central
London W2 6BY
UK

Email: kevin.smith@vodafone.com

Guenter Klas
Vodafone
One Kingdom Street, Paddington Central
Newbury RG14 2FN
UK

Email: kguenter.klas@vodafone.com

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

Copyright and License Notice

Table of Contents

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

```
                          Feedback
          |-----------------------------------------|
          |                                         |
          |                                         |
          |                                         V
      +----------+         tunnel            +-----------+
      |Egress    |========================== |Inress     |
      |(Exporter)|                           |(Collector)|
      +----------+                           +-----------+
```

(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

```
    Feedback    +-----------+
      --------->|Controller |####################
      |         |(Collector)|                   #
      |         +-----------+                   #
      |                                         #
   +----------+          tunnel            +-----V-+
   |Egress    | ===========================|Ingress|
   |(Exporter)|                            +-------+
   +----------+
```

(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 marks 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:

```
        +-------+                +------+
        |Ingress|                |Egress|
        +-------+                +------+
            |                       |
    +----------------+              |
    |cumulative count|              |
    +----------------+              |
            |                       |
            | <node id-i, ECN counts> |
            |------------------------->|
            |<node id-e, ECN counts>  |
            |<-------------------------|
            |                         |
            |                         |
```

(a) Direct model feedback procedure

```
     +----------+    +-------+                +------+
     |Controller|    |Ingress|                |Egress|
     +----------+    +-------+                +------+
          |              |                       |
          |      +----------------+              |
          |      |cumulative count|              |
          |      +----------------+              |
          |              |                       |
          |              | <node id-i, ECN counts> |
          |              |------------------------->|
          |              |                       |
          |                                      |
          |                                      |
          |       <node id-i, ECN counts>        |
          |       <node id-e, ECN counts>        |
          |<-------------------------------------|
          |                                      |
          |                                      |
          |                                      |
```

(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 Extentions

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

Statues: 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

Statues: 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

Statues: 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

Statues: 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

Statues: 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