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

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction .................................................. 3
   1.1. Contributing Authors ...................................... 3
   1.2. Terminology ................................................ 3
   1.3. Acronyms and Abbreviations ................................ 3
   1.4. Definitions ................................................ 4
   1.5. Assumptions and Considerations for the Solution .......... 4
2. Protocol ....................................................... 6
   2.1. Common Kind-Length-Value header ........................... 8
   2.2. Plain text mode Throughput Guidance Options .............. 10
   2.3. Encrypted mode ............................................ 11
   2.4. Nonce (Initialization Vector) .............................. 13
   2.5. Authentication ............................................ 14
3. Applicability to Video Delivery Optimization .................. 15
   3.1. Test Results ................................................ 15
4. Manageability considerations .................................... 16
5. Security considerations .......................................... 16
6. IANA considerations ............................................ 17
7. Acknowledgements ................................................ 18
8. References ...................................................... 18
   8.1. Normative References ....................................... 18
   8.2. Informative References ..................................... 18
Appendix A. .......................................................... 19
Authors’ Addresses ................................................. 19

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 in the RAN that signals to the TCP server the information on the (near-real time) throughput estimated to be available at the radio downlink interface.

1.5. Assumptions and Considerations for the Solution

This document specifies a solution protocol that is compliant 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, e.g HTTPS, is applied.
In particular, the Throughput Guidance Providers add 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. The proposed scheme is similar to 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]). Note, however, that 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 information element 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 downlink path for a given TCP connection. The information is sent to the uplink endpoint 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 Kind-Length-Value structure 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 and/or encryption capabilities. Throughput guidance is the conveyed information in this document. 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 resides on top of a radio network element see [Req_Arch_MTG_Exposure]).

Confidential information must be delivered in a secure way. The information can be provided as plain text in a secure and closed network. In other cases, the information should be authenticated and encrypted at the TCP-header level (between the Throughput Guidance Provider and the TCP server). An acceptable level of authentication and encryption (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 in such a case.

Two transfer modes are defined to deal with data confidentiality in this document; namely, plain-text mode and authenticated encryption mode. A third mode, authentication-only mode, is equally feasible. A third mode, authentication-only mode, is equally feasible and may
use TCP Authentication Option (TCP-AO) (see RFC 5935 [RFC5925]). We will describe the authentication-only mode in detail in future version of this draft. Both modes share a common Kind-Length-Value "option header" structure with a flag field separating the two cases.

2.1. Common Kind-Length-Value header

Mobile Throughput Guidance Signaling uses the common TCP options structure as in [RFC793] with experimental identifier as defined in [RFC6994]. To make Mobile Throughput Guidance Signaling extendible to different use cases a common Kind-Length-Value structure is defined below.

```
+------------------------------------------+
| Kind | Length | ExID | Flags | variable length data |
+------------------------------------------+
```

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.

Flags:

One byte of MTG protocol flag field as defined below.
Flag field of common Kind-Length-Value header

Figure 2

Seq:

Three-bit sequence number that maintains context across different packet types as defined by P- and T-bits below. The scope of the sequence number is to protect against packet reordering, not to provide a globally unique identifier or sequence number. The use of these bits are reserved for possible transfer mode extensions.

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.

Variable length data:

The variable length content (i.e. option data) in <type, value> format. The content depends of the transfer mode as defined in the following sections of this document. If the option data is fragmented across multiple headers the first fragment (marked with Frag=001 in the Flags-field) contains "Total Length of Data"-field that is the length of the variable data of MTG in all the fragments. Total Length of Data field is followed the content in <type, value>-format.

As an example for the use of the Flags-field, consider a cipher text of a single block. For it the T-bit is set to one, P-bit is set to
zero, Fragment and Seq-fields are zero in the Flags-field. In case the cipher text option cannot fit into a single TCP packet option, the cipher text is fragmented across multiple TCP headers. The first fragment has value Frag= 001, and the value is incremented for each subsequent fragment. The first fragment contains the "Total Length of Data"-field indicating the total length of the data to be fragmented. Last fragment is marked with all Frag-bits set to 1 (Frag= 111 for the last fragment). Therefore, the maximum number of fragments is seven. Details follow in the next sections.

2.2. Plain text mode Throughput Guidance Options

The plain text mode can be used in secure and closed networks or with information that has no confidentiality requirement. The plain text mode is made of one or more type-value pairs. The type determines the length of the following value.

Table of Type Value pairs of Throughput Guidance option data

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Unit of the type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput Guidance</td>
<td>1</td>
<td>2 bytes</td>
<td>Mbits/s</td>
</tr>
</tbody>
</table>

Table 1: MTG type-value pairs

The Type 1 element carries the actual throughput estimate in the 16-bit value field. The throughput value is encoded using a fixed-point number representation. 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. Throughput is expressed in Megabits per second.

The type-value pair elements are laid out consecutively in the header. At the end padding (i.e., the NO-OP TCP Option header with kind equal to 1, or the End of Option List TCP Option header with kind equal to 0) may be required to align the header size to the multiple of 4 bytes (required by the TCP standard). All bits in the Flag field are set to zero.
Kind, Length, ExID remains same as described in section 2.1. Options data constitutes the Flags and the variable length data. Flags: P- and T-bits set to zero

Layout of plain text option data in the TCP header options space.

Figure 3

2.3. Encrypted mode

Encryption requires authentication for integrity protection, as it is insecure to use encryption without it. Thus, the encrypted mode contains authentication as well. Encryption and authentication must use different keys. The following diagram shows the encryption process.

Encryption method

Figure 4

The encryption uses Advanced Encryption Standard (AES), 128 bits (16 bytes) block size, 128 bits (16 bytes) key size, Counter (CTR) block cipher mode. Integrity protection with CTR mode is MUST; this is provided via HMAC based message authentication (see Authentication section below).

The plaintext contains type-value pair elements of the variable length data. The plaintext is divided into blocks of 16 bytes. A
block of plain text MUST not exceed 16 bytes in a single run. Encryption takes a key (16 bytes), an IV or Nonce (16 bytes), the plain-text (at most 16 bytes) and produces a cipher text of 16 bytes. Note: multiple keys, at most 256, may be available (can be exchanged via an out-of-band key management mechanism such as Diffie-Hellman key exchange; this is out of scope of this document) for encryption key index. The keys MUST be different from those used for authentication.

The Nonce is 16 bytes. A unique Nonce is generated for each encrypted block. The same Initialization Vector, IV or Nonce MUST NOT be used with the same encryption key more than once. This is to be enforced by the Throughput Guidance Provider; otherwise security scheme will be broken.

The resulting cipher text is in blocks of 16 bytes. The cipher text blocks are packed into the option space together with the used Key Index in a following way if they fit into single option space of a single TCP header.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
--+-|Kind|Length|ExID|Flags|KeyIndex|first block of 16 bytes|
|--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Kind, Length, ExID remains same as described in section 2.1. Options data constitutes the Flags and the variable length data. Flags: Type of cipher text T-bit set to 1, only one block Frag= 000. Key Index is the index used in encryption

Cipher text layout in the TCP options without fragmentation

*Figure 5*

The flag field of the common option header indicates that the content is cipher text by having the T bit set to one. Since the ciphered block is not fragmented the Frag-bits of the flag field are set to zero (Frag= 000). (Use of Seq bits is left for later submissions). If there is not enough space to accommodate the 16 bytes in the option data, the data is fragmented.

If there are multiple cipher text blocks of 16 bytes, the flag field shows the type of the option being cipher text with the T-bit set to one, and by Frag-field showing the fragment number starting from 001 and incremented by one for each subsequent fragment of a packet of
the same type. For the last fragment, the Frag-field is always binary 111 to indicate the last fragment.

First fragment:

```
+-----------------------------------------------+
| Kind | Length|ExID|Flags| Total Length|KeyIndex|1. block |fragmented block |
+-----------------------------------------------+
```

Kind, Length, ExID remains same as described in section 2.1
Options data constitutes the Flags, Total Length, Key Index and the variable length data.
Flags: Type of cipher text T-bit = 1, Frag field = 001 first fragment
Total Length: total number of bytes of option data to be fragmented
Key Index is the index used in encryption

Second fragment if the last one:

```
+-----------------------------------------------+
| Kind | Length | ExID |Flags| Key Index | Rest of the fragmented block |
+-----------------------------------------------+
```

Kind, Length, ExID remains same as described in section 2.1
Options data constitutes the Flags, Key Index and the variable length data.
Flags: Type of cipher text T-bit = 1, Frag field = 111 last fragment, otherwise 010.
Total Length: total number of bytes in the fragments
Key Index is the index used in encryption

Cipher text layout extending to two consecutive headers

Figure 6

2.4. Nonce (Initialization Vector)

The 16 byte Nonce (or IV) is transmitted along with the cipher text to protect against de-synchronization between the encryption-decryption points.
2.5. Authentication

The authentication covers the cipher text, the Nonce (IV) and includes additional TCP protocol header fields to protect against replay attacks. The authentication uses HMAC codes (e.g., HMAC-SHA2-224), 128 bits (16 bytes) key size, 224 bits (28 bytes) digest size. Multiple keys (at most 256) for authentication with the same information receiver can be used. The keys MUST be different from those used for encryption. 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 28 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 authentication covers the TCP sequence number, ACK number, and TimeStamp (TSval, TSecr, not the possible 2 bytes of padding) fields of the TCP header as well as the Common Kind-Length-ExID-header with its data in all cipher text option and IV/Nonce option packets. (The Authentication type options itself cannot be covered by the authentication.)

The order in which the fields are included into the message authentication code is the following. From the TCP header: TCP Seq, ACK, TSval, TSecr. Followed by the following fields from the ciphered text: Kind, Length, ExID, Flags, Key Index, cipher text, and
from the IV/Nonce type of option packets TCP Seq, ACK, TSval, TSsecr
(note cipher text and IV/Nonce type of options may be in different
TCP packets) followed by Kind, Length, ExID, Flags, Key Index, Nonce/
IV.

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 environment Different network load scenarios
were taken into consideration. TCP Cubic was used in these tests and
the TG information was used by the TCP based video server to adjust
TCP congestion window only. The results below are based on data for
whole 2 days (23rd and 25th Feb 2015).

All network level metrics showed an average improvement of 30-60%, as detailed below:

- Reduction of end-to-end TCP RTT by 55-70%
- TCP retransmissions reduced by 30-45%
- Mean Client Throughput improved by 20-35%
- TCP packet loss reduced by 35-50%

The application-level metrics show an average improvement as detailed below:

- Click-to-play time reduced by 5-20%
- Average video resolution improvement by 5-20%
- Reduction in the number of format changes by 10 - 25%

These user experience improvements result in faster video time to play and are likely to result in longer battery life.

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 is considered confidential information and it 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 and encrypted at the TCP-header level (between the Throughput Guidance Provider and the TCP server).

Section 2 described how the TCP Header information can be signed and encrypted for security purposes. An out-of-band mechanism is currently used to agree upon the set of keys used to encrypt and
authenticate the messages exchanged between the endpoint and the network element that generates the throughput guidance headers.

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 and the encryption algorithm 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 encrypt the header and 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).

The impact of TCP Authentication Option (TCP-AO) with encrypted TCP segment payload [tcp-ao-encrypt] implies that the Throughput Guidance Provider functional element acts as a full back to back TCP proxy. This case is left for later stages as the work [tcp-ao-encrypt] is still at draft stage.

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


8.2. Informative References


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
Helsinki
FI

Phone: +358504839522
Email: hannu.flinck@nokia.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

Email: kevin.smith@vodafone.com
Network Address Translation (NAT) Behavioral Requirements Updates

draft-ietf-tsvwg-behave-requirements-update-01

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 NAPT44.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on August 22, 2015.
Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction .................................................. 3
   1.1. Scope .................................................. 3
   1.2. Terminology ............................................ 3
2. TCP Session Tracking ......................................... 3
   2.1. TCP Transitory Connection Idle-Timeout ................. 4
   2.2. TIME_WAIT State ....................................... 5
      2.2.1. Proposal: Apply RFC6191 and PAWS to NAT .......... 5
   2.3. TCP RST ................................................ 8
3. Port Overlapping behavior .................................... 8
4. Address Pooling Paired (APP) ................................ 8
5. EIF Security .................................................. 9
6. EIF Protocol Independence ................................... 9
7. EIF Mapping Refresh ......................................... 9
   7.1. Outbound Mapping Refresh and Error Packets ............. 10
8. EIM Protocol Independence ................................... 10
9. Port Parity ................................................... 10
10. Port Randomization .......................................... 10
11. IP Identification (IP ID) .................................. 11
12. ICMP Query Mappings Timeout ................................ 11
13. Hairpinning Support for ICMP Packets ...................... 11
14. IANA Considerations ...................................... 11
15. Security Considerations .................................... 11
16. Acknowledgements ......................................... 12
17. References ................................................ 12
   17.1. Normative References .................................. 12
   17.2. Informative References ................................ 13
Authors’ Addresses .............................................. 13
1. Introduction

[RFC4787], [RFC5382] and [RFC5508] greatly advanced NAT interoperability and conformance. But with widespread deployment and evolution of NAT more development and operational experience was acquired some areas of the original documents need further clarification or updates. This document provides such clarifications and updates.

1.1. Scope

This document focuses solely on NAPT44 and its goal is to clarify, fill gaps or update requirements of [RFC4787], [RFC5382] and [RFC5508].

It is out of the scope of this document the creation of completely new requirements not associated with the documents cited above. New requirements would be better served elsewhere and if they are CGN specific in an update to [RFC6888].

1.2. Terminology

The reader should be familiar with the terms defined in [RFC2663],[RFC4787],[RFC5382],and [RFC5508]

2. TCP Session Tracking

[RFC5382] specifies TCP timers associated with various connection states but does not specify the TCP state machine a NAPT44 should use as a basis to apply such timers. The TCP state machine depicted in Figure 1, adapted from [RFC6146], provides guidance on how TCP session tracking could be implemented - it is non-normative.
2.1. TCP Transitory Connection Idle-Timeout

[ RFC5382 ] :REQ-5 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. But the document does not clearly state if these can be configured separately.

This document clarifies that a NAT device SHOULD provide different knobs for configuring the open and closing idle timeouts. This
document further acknowledges that most TCP flows are very short (less than 10 seconds) [FLOWRATE][TCPWILD] and therefore a partially open timeout of 4 minutes might be excessive if security is a concern. Therefore, it MAY be configured to be less than 4 minutes in such cases. There also may be cases that a timeout of 4 minutes might be excessive. The case and the solution are written below.

2.2. TIME_WAIT State

The TCP TIME_WAIT state is described in [RFC0793]. The TCP TIME_WAIT state needs to be kept for 2MSL before a connection is CLOSED, for the reasons listed below:

1: In the event that packets from a session are delayed in the in-between network, and delivered to the end relatively later, we should prevent the packets from being transferred and interpreted as a packet that belongs to a new session.

2: If the remote TCP has not received the acknowledgment of its connection termination request, it will re-send the FIN packet several times.

These points are important for the TCP to work without problems.

[RFC5382] leaves the handling of TCP connections in TIME_WAIT state unspecified and mentions that TIME_WAIT state is not part of the transitory connection idle-timeout. If the NAT device honors the TIME_WAIT state, each TCP connection and its associated resources is kept for a certain period, typically for four minutes, which consumes port resources.

[RFC6191] explains that in certain situation it is necessary to reduce the TIME_WAIT state and defines such a mechanism using TCP timestamps and sequence numbers. When a connection request is received with a four-tuple that is in the TIME-WAIT state, the connection request may be accepted if the sequence number or the timestamp of the incoming SYN segment is greater than the last sequence number seen on the previous incarnation of the connection.

This document specifies that a NAT device should keep TCP connections in TIME_WAIT state unless it implements the proposal described in the following sub-section.

2.2.1. Proposal: Apply RFC6191 and PAWS to NAT

This section proposes to apply [RFC6191] mechanism at NAT. This mechanism MAY be adopted for both clients’ and remote hosts’ TCP active close.
Also, PAWS works to discard old duplicate packets at NAT. A packet can be discarded as an old duplicate if it is received with a timestamp or sequence number value less than a value recently received on the connection.

To make these mechanisms work, we should concern the case that there are several clients with nonsuccessive timestamp or sequence number values are connected to a NAT device (i.e., not monotonically increasing among clients). Two mechanisms to solve this mechanism and applying [RFC6191] and PAWS to NAT are described below. These mechanisms are optional.

2.2.1.1. Rewrite timestamp and sequence number values at NAT

Rewrite timestamp and sequence number values of outgoings packets at NAT to be monotonically increasing. This can be done by adopting following mechanisms at NAT.
A: Store the newest rewritten value of timestamp and sequence number as the "max value at the time".

B: NAT rewrite timestamp and sequence number values of incoming packets to be monotonically increasing.

When packets come back as replies from remote hosts, NAT rewrite again the timestamp and sequence number values to be the original values. This can be done by adopting following mechanisms at NAT.

C: Store the values of original timestamp and sequence number of packets, and rewritten values of those.

2.2.1.2. Split an assignable number of port space to each client

Adopt following mechanisms at NAT.

A: Choose clients that can be assigned ports.

B: Split assignable port numbers between clients.

Packets from other clients which are not chosen by these mechanisms are rejected at NAT, unless there is unassigned port left.

2.2.1.3. Resend the last ACK to the retransmitted FIN

We need to solve another scenario to make [RFC6191] work with NAT. In the case the remote TCP could not receive the acknowledgment of its connection termination request, the NAT device, on behalf of clients, resends the last ACK packet when it receives a FIN packet of the previous connection, and when the state of the previous connection has been deleted from the NAT. This mechanism MAY be used when clients starts closing process, and the remote host could not receive the last ACK.

2.2.1.4. Remote host behavior of several implementations

To solve the port shortage problem on the client side, the behavior of remote host should be compliant to [RFC6191] or the mechanism written in Section 4.2.2.13 of [RFC1122], since NAT may reuse the same 5 tuple for a new connection. We have investigated behaviors of OSes (e.g., Linux, FreeBSD, Windows, MacOS), and found that they implemented the server side behavior of the above two.
2.3. TCP RST

[RFC5382] leaves the handling of TCP RST packets unspecified. This document does not try standardize such behavior but clarifies based on operational experience that a NAT that receives a TCP RST for an active mapping and performs session tracking MAY immediately delete the sessions and remove any state associated with it. If the NAT device that performs TCP session tracking receives a TCP RST for the first session that created a mapping, it MAY remove the session and the mapping immediately.

3. Port Overlapping behavior

[RFC4787] [RFC5382]: REQ-1 Current RFCs specify a specific port overlapping behavior, i.e., that the external IP:port can be reused for connections originating from the same internal source IP:port irrespective of the destination. This is known as endpoint-independent mapping. This document clarifies that this port overlapping behavior can be extended to connections originating from different internal source IP:ports as long as their destinations are different. This known as EDM (Endpoint Dependent Mapping). The mechanism below MAY be one optional implement to NAT.

If destination addresses and ports are different for outgoing connections started by local clients, 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 enables NAT to work correctly in IP address resource limited network.

Discussions:

[RFC4787] and [RFC5382] requires "endpoint-independent mapping" at NAT, and port overlapping NAT cannot meet the requirement. This mechanism can degrade the transparency of NAT in that its mapping mechanism is endpoint-dependent and makes NAT traversal harder. However, if a NAT adopts endpoint-independent mapping together with endpoint-dependent filtering, then the actual behavior of the NAT will be the same as port overlapping NAT.

4. Address Pooling Paired (APP)

[RFC4787]: REQ-2 [RFC5382]: ND Address Pooling Paired behavior for NAT is recommended in previous documents but behavior when a public IPv4 run out of ports is left undefined. This document clarifies that if
APP is enabled new sessions from a subscriber that already has a mapping associated with a public IP that ran out of ports SHOULD be dropped. The administrator MAY provide a knob that allows a NAT device to starting using ports from another public IP when the one that anchored the APP mapping ran out of ports. This is trade-off between subscriber service continuity and APP strict enforcement. (Note, it is sometimes referred as ‘soft-APP’)

5. EIF Security

[RFC4787]:REQ-8 and [RFC5382]:REQ-3 End-point independent filtering could potentially result in security attacks from the public realm. In order to handle this, when possible there MUST be strict filtering checks in the inbound direction. A knob SHOULD be provided to limit the number of inbound sessions and a knob SHOULD be provided to enable or disable EIF on a per application basis. This is specially important in the case of Mobile networks where such attacks can consume radio resources and count against the user quota.

6. EIF Protocol Independence

[RFC4787]:REQ-8 and [RFC5382]:REQ-3 Current RFCs do not specify whether EIF mappings are protocol independent. In other words, if an outbound TCP SYN creates a mapping, it is left undefined whether inbound UDP packets destined to that mapping should be forwarded. This document specifies that EIF mappings SHOULD be protocol independent in order allow inbound packets for protocols that multiplex TCP and UDP over the same IP: port through the NAT and also maintain compatibility with stateful NAT64 RFC6146 [RFC6146]. But, the administrator MAY provide a configuration knob to make it protocol dependent.

7. EIF Mapping Refresh

[RFC4787]: REQ-6 [RFC5382]: ND The NAT mapping Refresh direction MAY have a "NAT Inbound refresh behavior" of "True" but it does not clarifies how this 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. This document clarifies that even when a NAT device has an inbound refresh behavior of TRUE, such packets SHOULD NOT refresh the mapping. Otherwise a simple attack of a packet every 2 minutes can keep the mapping indefinitely.
7.1.  Outbound Mapping Refresh and Error Packets

In the case of NAT outbound refresh behavior there are certain types of packets that should not refresh the mapping even if their direction is outbound. For example, if the mapping is kept alive by ICMP Errors or TCP RST outbound packets sent as response to inbound packets, these SHOULD NOT refresh the mapping.

8.  EIM Protocol Independence

[RFC4787] [RFC5382]: REQ-1 Current RFCs do not specify whether EIM are protocol independent. In other words, if a outbound TCP SYN creates a mapping it is left undefined whether outbound UDP can reuse such mapping and create session. On the other hand, Stateful NAT64 [RFC6146] clearly specifies three binding information bases (TCP, UDP, ICMP). This document clarifies that EIM mappings SHOULD be protocol dependent. A knob MAY be provided in order allow protocols that multiplex TCP and UDP over the same source IP and port to use a single mapping.

9.  Port Parity

A NAT devices MAY disable port parity preservation for dynamic mappings. Nevertheless, A NAT SHOULD support means to explicitly request to preserve port parity (e.g., [I-D.ietf-pcp-port-set]).

10.  Port Randomization

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)."
11. IP Identification (IP ID)

A NAT SHOULD handle the Identification field of translated IPv4 packets as specified in Section 9 of [RFC6864].

12. ICMP Query Mappings Timeout

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

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

13. Hairpinning Support for ICMP Packets

[RFC5508]:REQ-7 This requirement specifies that NAT devices enforcing Basic NAT MUST support traversal of hairpinned ICMP Query sessions. 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, NAT device can then translate to an internal IP. [RFC5508]:REQ-7 This requirement specifies that all NAT devices (i.e., Basic NAT as well as NAPT devices) MUST support the traversal of hairpinned ICMP Error messages. This requires NAT devices 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 that RFC.

14. IANA Considerations

This document does not require any IANA action.

15. Security Considerations

In the case of EIF mappings due to high risk of resource crunch, a NAT device MAY provide a knob to limit the number of inbound sessions spawned from a EIF mapping.

[I-D.ietf-tcpm-tcp-security] contains a detailed discussion of the security implications of TCP Timestamps and of different timestamp generation algorithms.
16. Acknowledgements

Thanks to Dan Wing, Suresh Kumar, Mayuresh Bakshi, Rajesh Mohan and Senthil Sivamular for review and discussions

17. References

17.1. Normative References


17.2. Informative References


Authors’ Addresses

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

Email: repenno@cisco.com

Simon Perreault
Viagenie
2875 boul. Laurier, suite D2-630
Quebec, QC G1V 2M2
Canada

Email: simon.perreault@viagenie.ca
Network Transport Circuit Breakers
draft-ietf-tsvwg-circuit-breaker-00

Abstract

This note explains what is meant by the term "network transport circuit breaker" (CB). It describes the need for circuit breakers when using network tunnels, and other non-congestion controlled applications. It also defines requirements for building a circuit breaker and the expected outcomes of using a circuit breaker within the Internet.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 29, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

Fairhurst Expires March 29, 2015
1. Introduction

A network transport Circuit Breaker (CB) is an automatic mechanism that is used to estimate congestion caused by a flow, and to terminate (or significantly reduce the rate of) the flow when persistent congestion is detected. 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.

The term "Circuit Breaker" originates in electricity supply, and has nothing to do with network circuits or virtual circuits. In such cases, a CB is intended as a protection mechanism of last resort. Under normal circumstances, a CB should not be triggered; it is designed to protect things when there is overload. Just as people do not expect the 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 CB principle can be used as a protection mechanism of last resort to avoid persistent congestion. Persistent congestion (also known as "congestion collapse") was a feature of the early Internet of the 1980s. This resulted in excess traffic starving other connection from access to the Internet. It was countered by the requirement to use congestion control (CC) by the Transmission Control Protocol (TCP) [Jacobsen88] [RFC1112]. These mechanisms operate in Internet hosts to cause TCP connections to "back off" during congestion. The introduction of CC in TCP (currently documented in [RFC5681] ensured the stability of the Internet, because it was able to detect congestion and promptly react. This worked well while TCP was by far the dominant traffic in the Internet, and most TCP flows were long-lived (ensuring that they could detect and respond to congestion before the flows terminated). This is no longer the case, and non-congestion controlled traffic, including many applications of 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 needs to be considered to avoid congestion collapse.

There are important differences between a transport circuit-breaker and a congestion-control method. Specifically, congestion control (as implemented in TCP, SCTP, and DCCP) needs to operate on the timescale on the order of a packet round-trip-time (RTT), the time from sender to destination and return. Congestion control methods may react to a single packet loss/marking and reduce the transmission rate for each loss or congestion event. The goal is usually to limit the maximum transmission rate that reflects the available capacity of a network path. These methods typically operate on individual traffic flows (e.g. a 5-tuple).

In contrast, CBs are recommended for non-congestion-controlled Internet flows and for traffic aggregates, e.g. traffic sent using a network tunnel. Later sections provide examples of cases where circuit-breakers may or may not be desirable.

A CB needs to measure (meter) the traffic to determine if the network is experiencing congestion and must be designed to trigger robustly when there is persistent congestion. This means 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 transports (or applications) to adjust their rate following congestion, and for the network load to stabilise after any adjustment. A CB trigger will often be based
on a series of successive sample measurements taken over a reasonably
long period of time. This is to ensure that a CB does not
accidentally trigger following a single (or even successive)
congestion events (congestion events are what triggers congestion
control, and are to be regarded as normal on a network link operating
near its capacity). Once triggered, a control function needs to
remove traffic from the network, either disabling the flow or
significantly reducing the level of traffic. This reaction provides
the required protection to prevent persistent congestion being
experienced by other flows that share the congested part of the
network path.

1.1. Types of Circuit-Breaker

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

- Fast-Trip Circuit Breakers: The relatively short timescale used by
  this form of circuit breaker is intended to protect a flow or
  related group of flows.

- Slow-Trip Circuit Breakers: This circuit breaker utilises a longer
timescale and is designed to protect traffic aggregates.

- Managed Circuit Breakers: Utilise the operations and management
  functions that may 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 advise on these two topics.

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 transport circuit breaker involves communication between an ingress point (a sender) and an egress point (a receiver) of a network flow. A simple picture of CB operation is provided in figure 1. This shows a set of routers (each labelled R) connecting a set of endpoints. A CB is used to control traffic passing through a subset of these routers, acting between an ingress and an egress point. In some cases the ingress and egress may be in one or both endpoints, in other cases they will be in the network, for example one expected use would be at the ingress and egress of a tunnel service.

Figure 1: A CB controlling the part of the end-to-end path between an ingress point and an egress point.

The set of components needed to implement a circuit breaker are:

1. An Ingress meter (at the sender or tunnel ingress) records the number of packets/bytes sent in each measurement interval. This measures the offered network load. The measurement interval could be every few seconds.

2. An Egress meter (at the receiver or tunnel egress) records the number/bytes received in each measurement interval. This measures the supported load and may utilise other signals to detect the effect of congestion (e.g. loss/marking experienced over the path).
3. The measured values at the ingress and egress are communicated to the CB Measurement function. This may use several methods including: Sending return measurement packets from a receiver to a trigger function at the sender; An implementation using Operations, Administration and Management (OAM), or another in-band signalling datagram to send to the trigger function; It could also be implemented purely as a control plane function using a software-defined network controller.

4. The Measurement function 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 counter values. Note that accurate measurement intervals are not typically important, since isolated loss events need to be disregarded.)

5. A Trigger function determines if the measurements indicate persistent congestion. This defines an appropriate threshold for determining there is persistent congestion between the ingress and egress (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 transport CB is triggered when the threshold is exceeded in multiple measurement intervals (e.g. 3 successive measurements). This design needs to be robust to single or spurious events triggering a reaction.

6. A Reaction that is applied at the Ingress when the CB is triggered. This seeks to automatically remove the traffic causing persistent congestion.

7. The CB also triggers when it does not receive both sender and receiver measurements, since this also could indicate a loss of control packets (also a symptom of heavy congestion or inability to control the load).

3.2. Requirements for implementing a CB

The requirements for implementing a CB are:

- There MUST be a control path from the Ingress meter and the Egress meter to the point of measurement. The CB MUST trigger if this control path fails. That is, the feedback indicating a congested period is designed so that the CB is triggered when it fails to receive measurement reports that indicate an absence of congestion, rather than relying on the successful transmission of a "congested" signal back to the sender. (The feedback signal could itself be lost under congestion collapse).
A CB MUST define a measurement period over which the receiver measures the level of congestion. 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. If Explicit Congestion Notification (ECN) is enabled [RFC3168], an egress meter MAY also count the number of ECN congestion marks/event per measurement interval, but even if ECN is used, loss MUST still be measured, since this better reflects the impact of persistent congestion. The type of CB will determine how long this measurement period needs to be. The minimum time must be significantly longer than the time that current CC algorithms need to reduce their rate following detection of congestion (i.e. many path RTTs).

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

A CB is REQUIRED to define a period over which the Trigger uses the collected measurements.

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. triggered when persistent congestion is detected in 3 measurements within the triggering interval).

A triggered CB MUST react decisively by disabling (or significantly reducing) traffic at the source (e.g. tunnel ingress). The CB SHOULD be constructed so that it does not trigger under light or intermittent congestion, with a default response to a trigger that disables all traffic that contributed to congestion.

Some circuit breaker designs use a reaction that reduces, rather that disables, the flows it control. This response MUST be much more severe than that of a CC algorithm, because the CB reacts to more persistent congestion and operates over longer timescales. A CB that reduces the rate of a flow, MUST continue to monitor the level congestion and MUST further reduce the rate if the CB is again triggered.

The reaction to a triggered CB MUST continue for a period of time of at least the triggering interval. Manual operator intervention will usually be required to restore the flow. If an automated response is needed to reset the trigger, then this MUST NOT be immediate. The design of this release 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.

- When a CB is triggered, it 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.

4. Examples of Circuit Breakers

There are multiple types of CB that may be defined for use in different deployment cases. This section provides examples of different types of circuit breaker:

4.1. A Fast-Trip Circuit Breaker

A fast-trip circuit breaker is the most responsive form of CB. It has a response time that is only slightly larger than that of the traffic it controls. It is suited to traffic with well-understood characteristics. It is not be suited to arbitrary network traffic, since it may prematurely trigger (e.g. when multiple congestion-controlled flows lead to short-term overload).

4.1.1. A Fast-Trip Circuit Breaker for RTP

A set of fast-trip CB 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 CB is therefore implemented as a fail-safe.

The sender monitors reception of RTCP Reception Report (RR or XRR) packets that convey reception quality feedback information. This is used to measure (congestion) loss, possibly in combination with ECN [RFC6679].

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

1. An RTP CB triggers on reported lack of progress.
2. An RTP CB triggers when no receiver reports messages are received.
3. An RTP CB uses a TFRC-style check and set a hard upper limit to the long-term RTP throughput (over many RTTs).
4. An RTP CB 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.

4.2. A Slow-trip Circuit Breaker

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

One example where a slow-trip CB 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, IFR), see [RFC5405] section 3.1.3. A use case is where tunnels are deployed in the general Internet (rather than "controlled environments" within an ISP or Enterprise), especially when the tunnel may need to cross a customer access router.

4.3. A Managed Circuit Breaker

A managed CB 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 congestion.

4.3.1. 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. MPLS) infrastructure, then it may be expected that the Pseudo-Wire (PW) would not experience congestion, because a flow is not expected to either increase (or decrease) their rate. If instead Pseudo-Wire traffic is multiplexed with other traffic over the general Internet, it could experience congestion. [RFC4553] states: "If SAToP FWs 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 should 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 the PW was likely have a serious problem, possibly making the service non-compliant. At this point triggering the CB would remove the traffic prevent undue impact congestion-responsive traffic (e.g., TCP). Part of the rationale, was that high loss ratios typically indicated that something was "broken" and should have already resulted in operator intervention, and should trigger this intervention. An operator-based response provides 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 may require the trigger to be sent to a third location (e.g. a network operations centre, NOC) responsible for operation of the tunnel ingress, rather than the tunnel ingress itself.

5. Examples where circuit breakers may not be needed.

A CB is not required for a single CC-controlled flow using TCP, SCTP, TFRC, etc. In these cases, the CC methods are designed to prevent congestion collapse.

XX NOTE: Comments on this section are particularly welcome to establish clearer understanding of the operational conditions under which circuit breakers should or must be deployed.

5.1. CBs and uni-directional Traffic

A CB can be used to control uni-directional UDP traffic, providing that there is a control path to connect the functional components at the Ingress and Egress. This control path can exist in networks for which the traffic flow is purely unidirectional (e.g. a multicast stream that sends packets across an Internet path).

A one-way physical link may have no associated control path, and therefore cannot be controlled using an automated process. This could be managed by policing traffic to ensure it does not exceed the available capacity. Supporting this type of traffic in the general Internet requires operator monitoring to detect and respond to persistent congestion or the use of dedicated capacity - e.g. Using per-provisioned MPLS services, RSVP, or admission-controlled Differentiated Services.
5.2. CBs over pre-provisioned Capacity

One common question is whether a CB 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 should not result in congestion. A CB will hence only be triggered when there is non-compliant traffic. It could be argued that this event should never happen - but it may also be argued that the CB equally should never be triggered. If a CB were to be implemented, it would provide an appropriate response should this persistent congestion occur in an operational network.

5.3. CBs with CC 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 [RFC5405]. A question therefore arises when people deploy a tunnel that is thought to only carry an aggregate of TCP (or some other CC-controlled) traffic: Is there advantage in this case in using a CB?

For sure, traffic in a such a tunnel will respond to congestion. However, the answer to the question may not be obvious, because the overall traffic formed by an aggregate of flows that implement a CC mechanism does not necessarily prevent congestion collapse. For instance, most CC mechanisms require long-lived flows to react to reduce the rate of a flow, an aggregate of many short flows may result in many terminating before they experience congestion. It is also often impossible for a tunnel service provider to know that the tunnel only contains CC-controlled traffic (e.g. Inspecting packet headers may not be possible). The important thing to note is that if the aggregate of the traffic does not result in persistent congestion (impacting other flows), then the CB will not trigger. This is the expected case in this context - so implementing a CB will not reduce performance of the tunnel, but offers protection should persistent congestion occur.

6. Security Considerations

This section will describe security considerations, if any.

7. IANA Considerations

This document makes no request from IANA.
8. Acknowledgments

There are many people who have discussed and described the issues that have motivated this draft. Contributions and comments are appreciated, including: Lars Eggert, Colin Perkins, David Black, Matt Mathis.

9. Revision Notes

RFC-Editor: Please remove this section prior to publication

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.

10. References

10.1. Normative References


10.2. Informative References


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
DiffServ interconnection classes and practice
draft-ietf-tsvwg-diffserv-intercon-01

Abstract

This document proposes a limited set of DiffServ PHBs and codepoints to be applied at (inter)connections of two separately administered and operated networks. Many network providers operate MPLS using Treatment Aggregates for traffic marked with different DiffServ PHBs, and use MPLS for interconnection with other networks. This document offers a simple interconnection approach that may simplify operation of DiffServ for network interconnection among providers that use MPLS.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

RFC2474 specifies the DiffServ Codepoint Field [RFC2474]. Differentiated treatment is based on the specific DSCP. Once set, it may change. If traffic marked with unknown or unexpected DSCPs is received, RFC2474 recommends forwarding that traffic with default (best effort) treatment without changing the DSCP markings. Many networks do not follow this recommendation, and instead remark unknown or unexpected DSCPs to the zero DSCP upon receipt for consistency with default (best effort) forwarding in accordance with
the guidance in RFC 2475 [RFC2474] to ensure that appropriate DSCPs are used within a DiffServ domain.

This document is motivated by requirements for IP network interconnection with DiffServ support among providers that operate MPLS in their backbones, but is applicable to other technologies. The operational simplifications and methods in this document help align IP DiffServ functionality with MPLS limitations; further, limiting DiffServ to a small number of Treatment Aggregates can enable network traffic to leave a network with the same DSCPs that it was received with, even if a different DSCP is used within the network, thus providing an opportunity to extend consistent QoS treatment across network boundaries.

In isolation, use of standard interconnection PHBs and DSCPs may appear to be additional effort for a network operator. The primary offsetting benefit is that the mapping from or to the interconnection PHBs and DSCPs is specified once for all of the interconnections to other networks that can use this approach. Otherwise, the PHBs and DSCPs have to be negotiated and configured independently for each network interconnection, which has poor scaling properties. Further, end-to-end QoS treatment is more likely to result when an interconnection code point scheme is used because traffic is remarked to the same PHBs at all network interconnections. This document envisions one-to-one DSCP remarking at network interconnections (not in DSCP to one DSCP remarking).

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

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

1.2. Applicability Statement

This document is primarily applicable to use of Differentiated Services for interconnection traffic between networks, and in particular to interconnection of MPLS-based networks. The approach described in this document is not intended for use within the interconnected (or other) networks, where the approach specified in RFC 5127 (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 to transport plain IP traffic terminating within or transiting through the receiving domain. Transit 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.

1.3. Document Organization

This document is organized as follows: section 2 reviews the MPLS Short Pipe tunnel model for Diffserv Tunnels [RFC3270]; effective support for that model is a crucial goal of this document. Section 3 provides background on RFC 5127’s approach to traffic class aggregation within a Diffserv network domain and explains why this document uses a somewhat different approach. Section 4 introduces Diffserv interconnection Treatment Aggregates, plus the PHBs and DSCPs that are mapped to these Treatment Aggregates. Further, section 4 discusses treatment of non-tunneled and tunneled IP traffic and MPLS VPN QoS aspects. Finally Network Management PHB treatment is described. Annex B describes the impact of the MPLS Short Pipe model (penultimate hop popping) on QoS related IP interconnections.

2. MPLS and the Short Pipe tunnel model

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

RFC 2474’s recommendation to forward traffic with unrecognized DSCPs with Default (best effort) service without rewriting the DSCP has
proven to be a poor operational practice. Network operation and management are simplified when there is a 1-1 match between the DSCP marked on the packet and the forwarding treatment (PHB) applied by network nodes. When this is done, CS0 (the all-zero DSCP) is the only DSCP used for Default forwarding of best effort traffic, so a common practice is to use CS0 to remark traffic received with unrecognized or unsupported DSCPs at network edges.

MPLS networks are more subtle in this regard, as it is possible to encode the provider’s DSCP in the MPLS TC field and allow that to differ from the PHB indicated by the DSCP in the MPLS-encapsulated IP packet. That would allow an unrecognized DSCP to be carried edge-to-edge over an MPLS network, because the effective DSCP used by the MPLS network would be encoded in the MPLS label TC field (and also carried edge-to-edge); this approach assumes that a provider MPLS label with the provider’s TC field is present at all hops within the provider’s network.

The Short Pipe tunnel model and PHP violate that assumption because PHP pops and discards the MPLS provider label carrying the provider’s TC field. That discard occurs one hop upstream of the MPLS tunnel endpoint (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; propagating another DSCP edge-to-edge requires an IP tunnel of some form. See Annex B for a more detailed discussion.

If transport of a large number (much greater than 4) DSCPs is required across a network that supports this DiffServ interconnection scheme, a tunnel or VPN can be provisioned for this purpose, so that the inner IP header carries the DSCP that is to be preserved not to be changed. From a network operations perspective, the customer equipment (CE) is the preferred location for tunnel termination, although a receiving domains Provider Edge router is another viable option.

3. Relationship to RFC 5127

This document draws heavily upon RFC 5127’s approach to aggregation of DiffServ traffic classes for use within a network, but there are some important differences caused by the characteristics of network interconnects.
3.1. RFC 5127 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 meet its Service Level Agreements (SLAs). 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.

RFC 5127 provides an example aggregation of DiffServ service classes into 4 Treatment Aggregates. A small number of aggregates are used because:

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

- There should be unused codes for interconnection purposes. This leaves space for future standards, for private bilateral agreements and for local use PHBs and DSCPs.

- Migrations from one code point scheme to another may require spare QoS code points.

RFC 5127 also follows RFC 2474 in recommending transmission of DSCPs through a network as they are received at the network edge.

3.2. Differences from RFC 5127

Like RFC 5127, this document also uses four traffic aggregates, but differs from RFC 5127 in three important ways:

- It follows RFC 2475 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 when 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. As discussed in Section 2 above, the MPLS Short Pipe tunnel model effectively requires use of a DSCP that is locally valid for the network involved, leading to this remarking approach.
It treats network control traffic as a special case. Within a network, the CS6 DSCP is used for local network control traffic (routing protocols and OAM traffic that is essential for 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 traffic) 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; it is more likely to be specifically configured (e.g., most networks impose an exchange of BGP for obvious reasons). See Section 4.2 for further discussion.

Because network control traffic is treated as a special case, a fourth traffic aggregate is defined for use at network interconnections to replace the Network Control aggregate in RFC 5127. Network Control traffic may still be exchanged across network interconnections as further discussed in Section 4.2.

### 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 may involve remarking for the interconnection; such remarking is part of the DiffServ Architecture [RFC2475], at least for the network edge nodes involved in interconnection. This draft proposes a standard interconnection set of 4 Treatment Aggregates with well-defined DSCPs to be aggregated by them. A sending party remarks DSCPs from internal schemes to the interconnection code points. The receiving party remarks DSCPs to her internal scheme. The set of DSCPs and PHBs supported across the two interconnected domains and the treatment of PHBs and DSCPs not recognized by the receiving domain should be part of the interconnect SLA.

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

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

In contrast, VoIP is emerging as a valuable and important class of network traffic for which network-provided QoS is crucial, as even minor glitches are immediately apparent to the humans involved in the conversation.

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

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

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

**Bulk Real-Time Treatment Aggregate:** This Treatment Aggregate is designed to transport PHB AF41, DSCP 100 010 (the other AF4 PHB group PHBs and DSCPs may be used for future extension of the set of DSCPs carried by this Treatment Aggregate). This Treatment Aggregate is designed to transport the portions of RFC 5127’s Real Time Treatment Aggregate, which consume large amounts of bandwidth, namely Broadcast Video, Real-Time Interactive and Multimedia Conferencing. The treatment aggregate should be configured with a rate queue (which is in line with RFC 4594 for the mentioned traffic classes). As compared to RFC 5127, the number of DSCPs has been reduced to one (initially). The proposed queuing mechanism is in line with RFC4594 definitions for Broadcast Video and Real-Time
Interactive. If need for three-color marked Multimedia Conferencing traffic arises, AF42 and AF43 PHBs may be added.

**Assured Elastic Treatment Aggregate**

This Treatment Aggregate consists of the entire AF3 PHB group AF3, i.e., DSCPs 011 010, 011 100 and 011 110. As compared to RFC5127, just the number of DSCPs, which has been reduced. This document suggests to transport signaling marked by AF31. RFC5127 suggests to map Network Management traffic into this Treatment Aggregate, if no separate Network Control Treatment Aggregate is supported (for a more detailed discussion of Network Control PHB treatment see section 3.2). GSMA IR.34 proposes to transport signaling traffic by AF31 too.

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

RFC 4594’s Multimedia Streaming class has not been mapped to the above scheme. By the time of writing, the most popular streaming applications use TCP transport and adapt picture quality in the case of congestion. These applications are proprietary and still change behaviour frequently. Currently, the Bulk Real-Time Treatment Aggregate or the Assured Elastic Treatment Aggregate may be a reasonable match. NOTE: This paragraph would benefit from WG review and discussion.

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

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

```
Provider-O
RFC5127
| +----------+
| | AF21, AF22 |
Provider-W
GSMA 34.1
| +----------+
| CS3, CS2 |
```
Rtr PrO | Rtr PrW | Rtr Pr:
DiffServ
| AF31, AF32 | AF31, AF32
Intercon
| V

Provider-T domain
MPLS TC 2
DSCP rew.
AF21, AF22

Local DSCPs Provider-T
V
AF21, AF22

RtrDst:
AF21, AF22

DiffServ
AF31, AF32

Intercon
AF31, AF32

CS4, CS3

Provider-F
GSM IR.34
DiffServ Intercon example

Figure 1

Providers only need to deploy internal DSCP to DiffServ Intercon DSCP mappings to exchange traffic in the desired classes. Provider W has decided that the properties of his internal classes CS3 and CS2 are best met by the DiffServ Intercon Assured Elastic Treatment Aggregate, PHBs AF31 and AF32 respectively. At the outgoing peering interface connecting provider W with provider T the former’s peering router remarks CS3 traffic to AF31 and CS2 traffic to AF32. The domain internal PHBs of provider T that meet the requirements of DiffServ Intercon Assured Elastic Treatment Aggregate are AF2x. 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 TC value 2 for this aggregate. Traffic received with AF32 is similarly remarked to AF22, but uses the same MPLS TC for the Treatment Aggregate, i.e. TC 2. At the penultimate MPLS node, the top MPLS label is removed. The packet should be forwarded as determined by the incoming MPLS TC. The peering router connecting domain T with domain F classifies the packet by it’s domain T internal DSCP AF21 for the DiffServ Intercon Assured Elastic Treatment Aggregate. As it leaves domain T on the interface to domain F, this causes the packet to be remarked to AF31. The peering router of domain F classifies the packet for domain F internal PHB CS4, as this is the PHB with properties matching DiffServ Intercon’s Assured Elastic Treatment Aggregate. Likewise, AF21 traffic is remarked to AF32 by the peering router of domain T when leaving it and from AF32 to CS3 by domain F’s peering router when receiving it.

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

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

See section 4.1 for further discussion of this and DSCP transparency in general.
RFC2575 states that Ingress nodes must condition all other inbound traffic to ensure that the DS codepoints are acceptable; packets found to have unacceptable codepoints must either be discarded or must have their DS codepoints modified to acceptable values before being forwarded. For example, an ingress node receiving traffic from a domain with which no enhanced service agreement exists may reset the DS codepoint to the Default PHB codepoint. As a consequence, an interconnect SLA needs to specify not only the treatment of traffic that arrives with a supported interconnect DSCP, but also the treatment of traffic that arrives with unsupported or unexpected DSCPs.

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

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

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

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

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

The key requirement is that the network ingress interconnect DSCP be restored at network egress, and a key observation is that this is only feasible in general for a small number of DSCPs.

4.2. Treatment of Network Control traffic at carrier interconnection interfaces

As specified by RFC4594, section 3.2, Network Control (NC) traffic marked by CS6 is to be expected at 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 RFC 4594. A specific example is that some CS6 traffic exchanged across carrier interconnections is terminated at the network ingress node, e.g. if BGP is running between two routers on opposite ends of an interconnection link; in this case the operators would enter into a bilateral agreement to use CS6 for that BGP traffic.

The end-to-end QoS discussion in the previous section (4.1) is generally inapplicable to network control traffic - network control traffic is generally intended to control a network, not be transported across it. One exception is that network control traffic makes sense for a purchased transit agreement, and preservation of 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 transiting traffic, including transiting network control traffic. Use of an IP tunnel is suggested in order to reduce the risk of CS6 markings on transiting network control traffic being interpreted by the network providing the transit.

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

Interconnecting carriers should specify treatment of CS6 marked traffic received at a carrier interconnection which is to be forwarded beyond the ingress node. An SLA covering the following cases is recommended when a provider wishes to send CS6 marked traffic across an interconnection link which isn’t terminating at the interconnected ingress node:

Geib & Black          Expires September 10, 2015          [Page 13]
Internet-Draft              Diffserv Intercon                 March 2015

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

o classification of traffic which is network control traffic for the sending domain only. This traffic should be classified for a PHB offering similar properties as the NC class (e.g. AF31 as specified by this document). As an example GSMA IR.34 proposes an Interactive class / AF31 to carry SIP and DIAMETER traffic. While this is service control traffic of high importance to the interconnected Mobile Network Operators, it is certainly not Network Control traffic for a fixed network providing transit between such operators, and hence should not receive CS6 treatment in such a network.

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

5. Acknowledgements

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

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

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

8. References

8.1. Normative References


8.2. Informative References


Appendix A. Appendix A Carrier interconnection related DiffServ aspects

NOTE: This Appendix is likely to be deleted in the next version of this draft. The authors would appreciate comments on the value (or lack thereof) of this text.

This appendix provides a general discussion of PHB and DSCP mapping at IP interconnection interfaces.

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

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

o Support the PHB with a differing DSCP, and the DSCP of the sending
domain is used for another PHB.

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

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

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

o The received DSCP is available for usage within the domain.

RFC2474 suggests transporting packets received with unrecognized
DSCPs by the Default PHB and not changing the DSCP as received. Also
if a particular DSCP is unused within a domain, the network may
subsequently change its QoS design and assign a PHB to a formerly
unused DSCP, making transparent transport of that DSCP as an unknown
DSCP with the Default PHB no longer possible. Remarking to another
DSCP apart from the Default PHBs DSCP does not seem to be a good
option in the latter case, as it’s not clear which other DSCP should
be used. If a domain interconnects with many other domains, the
concerns discussed here may have to be dealt with many times.

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

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

DiffServ Intercon proposes a set of PHBs and corresponding DSCPs at
interconnection points. A PHB to DSCPs correspondence is specified
for interconnection interfaces. Supported PHBs should be available
end-to-end, but domain internal DSCPs may change end-to-end, although
they are restored at network interconnection points.
Appendix B. Appendix B 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 non-tunneled IPv4 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 tunneled IPv4 traffic, only the outer tunnel header is involved in 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.

Non-tunneled IPv6 traffic as well as Layer 2 and Layer 3 VPN traffic all use an additional MPLS label; forwarding within an MPLS network is based on that label, as opposed to the outer IP header.

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

Beyond RFC3270’s suggestions that the Short Pipe Model is only applicable to VPNs, current network structures also use it to transport non tunneled IPv4 traffic. This is shown in 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 QoS behavior.

Without DiffServ-Intercon treatment, the traffic is likely to leave each network marked with network-internal DSCP. DSCP_send of the figure above is remarked to the receiving network’s DiffServ scheme.
It leaves the domain marked by the domains 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 those 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
  | ^^^^^^  IPv4, DSCP_d

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

Appendix C. Change log

00 to 01 Added terminology and references. Added details and information to interconnection class and codepoint scheme. Editorial changes.
01 to 02  Added some references regarding related work. Clarified class definitions. Further editorial improvements.

02 to 03  Consistent terminology. Discussion of Network Management PHB at interconnection interfaces. Editorial review.

03 to 04  Again improved terminology. Better wording of Network Control PHB at interconnection interfaces.

04 to 05  Large rewrite and re-ordering of contents.

05 to 06  Description of IP and MPLS related requirements and constraints on DSCP rewrites.

06 to 07  Largely rewrite, improved match and comparison with RFCs 4594 and 5127.

07 to 08  Added Annex A and B which where forgotten when putting together -07

Authors' Addresses

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

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

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

Phone: +1 (508) 293-7953
Email: david.black@emc.com
Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP
draft-ietf-tsvwg-ecn-encap-guidelines-02

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 27, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ............................................. 3
   1.1. Scope ............................................. 5
2. Terminology ............................................. 6
3. Guidelines in All Cases ................................. 7
4. Modes of Operation ..................................... 7
   4.1. Feed-Forward-and-Up Mode .......................... 8
   4.2. Feed-Up-and-Forward Mode .......................... 10
   4.3. Feed-Backward Mode ................................ 10
   4.4. Null Mode ......................................... 12
5. Feed-Forward-and-Up Mode: Guidelines for Adding Congestion Notification ................................... 12
   5.1. IP-in-IP Tunnels with Tightly Coupled Shim Headers ... 13
   5.2. Wire Protocol Design: Indication of ECN Support ........ 13
   5.3. Encapsulation Guidelines ............................ 15
   5.4. Decapsulation Guidelines ............................ 17
   5.5. Sequences of Similar Tunnels or Subnets ............... 18
   5.6. Reframing and Congestion Markings .................... 19
6. Feed-Up-and-Forward Mode: Guidelines for Adding Congestion Notification ................................... 19
7. Feed-Backward Mode: Guidelines for Adding Congestion Notification ................................... 21
8. IANA Considerations (to be removed by RFC Editor) .......... 22
9. Security Considerations .................................. 22
10. Conclusions .............................................. 22
11. Acknowledgements ....................................... 23
12. Comments Solicited ..................................... 23
13. References .............................................. 23
   13.1. Normative References ................................. 23
   13.2. Informative References ............................... 24
Appendix A. Outstanding Document Issues ...................... 27
Appendix B. Changes in This Version (to be removed by RFC Editor) ........................................... 27
Authors’ Addresses .......................................... 29
1. Introduction

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

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

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

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

- 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, this has often been more due to the inability of the original TCP protocol to saturate the links. For many years, fixes such as window scaling [RFC1323] proved hard to deploy. But now that modern
operating systems are finally capable of saturating interior links, even the buffers of well-provisioned interior switches will need to signal episodes of queuing.

Propagation of ECN is defined for MPLS [RFC5129], and is being defined for TRILL [I-D.ietf-trill-rfc7180bis], but it remains to be defined for a number of other subnetwork technologies.

Similarly, ECN propagation is yet to be defined for many tunnelling protocols. [RFC6040] defines how ECN should be propagated for IP-in-IP [RFC2003] and IPSec [RFC4301] tunnels. However, as Section 9.3 of RFC3168 pointed out, ECN support will need to be defined for other tunnelling protocols, e.g. L2TP [RFC2661], GRE [RFC1701], [RFC2784], PPTP [RFC2637] and GTP [GTPv1], [GTPv1-U], [GTPv2-C].

Incremental deployment is the most tricky 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 deteriorate further. However, to support congestion marking below the IP layer, it is not sufficient to only check that the two end-points support ECN; correct operation also depends on the decapsulator at each subnet egress faithfully propagating congestion notifications to the higher layer. Otherwise, a legacy decapsulator might silently fail to propagate any ECN signals from the outer to the forwarded header. Then the lost signals would never be detected and again congestion would deteriorate further. The guidelines given later require protocol designers to carefully consider incremental deployment, and suggest various safe approaches for different circumstances.

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

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

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

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

1.1. Scope

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

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

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

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

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

2. Terminology

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

Further terminology used within this document:

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

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

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

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

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

Outer header: The header added to encapsulate a PDU.

Inner header: The header encapsulated by the outer header.

Outgoing header: The header forwarded by the decapsulator.

CE: Congestion Experienced [RFC3168]
ECT: ECN-Capable Transport [RFC3168]

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

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

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

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 or policing nodes). Note the term "a function capable of controlling the load" deliberately includes a transport that doesn’t actually control the load but ideally it ought to (e.g., a sending application without congestion control that uses UDP).

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

3. Guidelines in All Cases

RFC 3168 specifies that the ECN field in the IP header is intended to be marked by active queue management algorithms. Any congestion notification from an algorithm that does not conform to the recommendations in [I-D.ietf-aqm-recommendation] MUST NOT be propagated from a lower layer into the ECN field in IP.

4. Modes of Operation

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

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

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

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

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

4.1. Feed-Forward-and-Up Mode

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

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

This mode is illustrated in Figure 1. Along the middle of the figure, layers 2, 3 & 4 of the protocol stack are shown, and one packet is shown along the bottom as it progresses across the network from source to destination, crossing two subnets connected by a router, and crossing two switches on the path across each subnet. Congestion at the output of the first switch (shown as *) leads to a congestion marking in the L2 header (shown as C in the illustration of the packet). The chevrons show the progress of the resulting congestion indication. It is propagated from link to link across the subnet in the L2 header, then when the router removes the marked L2 header, it propagates the marking up into the L3 (IP) header. The router forwards the marked L3 header into subnet 2, and when it adds a new L2 header it copies the L3 marking into the L2 header as well,
as shown by the ‘C’ s in both layers (assuming the technology of
subnet 2 also supports explicit congestion marking).

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

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

Figure 1: Feed-Forward-and-Up Mode

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

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

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

```
+---+        layer: 2 3 4 header
|  <|<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Packet V <<<<<<<<<<<<<|<<|L4
|   |                         +---+                         | ^ |
|   | . . .  >>>> Packet U >>>|>>>|>>> Packet U >>>>>>>>>> |>^ |L3
|   |     +--^+     +---+     |   |     +---+     +---+     |   |
|___|_____|___|_____|___|_____|___|_____|___|_____|___|_____|___|
source          subnet E      router       subnet F         dest
```

Figure 2: Feed-Up-and-Forward Mode

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

4.3. Feed-Backward Mode

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

For instance, for the available bit-rate (ABR) service in ATM, the relative rate mechanism was one of the more popular mechanisms for managing traffic, tending to supersede earlier designs. In this approach ATM switches send special resource management (RM) cells in both the forward and backward directions to control the ingress rate of user data into a virtual circuit. If a switch buffer is approaching congestion or 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.

For feedback ATM sends control cells back to the ingress (see Figure 3).

\[\text{source} \quad \text{subnet G} \quad \text{router} \quad \text{subnet H} \quad \text{dest} \]

\[\begin{array}{cccc}
4 & 3 & 2 & 4 \\
4 & 3 & 2 & 4 \\
4 & 3 & 4 & 3 \\
\end{array}\]

\[\text{later data packet (W)} \]

\[\text{earlier data packet (U)} \]

\[\text{layer: 4 3 2 4 3 2} \]

\[\text{header} \]

\[\text{Feedback control cell/frame (V)} \]

\[\text{layer: 4 3 2 4 3 2} \]

\[\text{header} \]

\[\text{Figure 3: Feed-Backward Mode} \]

ATM’s feedback approach doesn’t fit well when layered beneath IP’s feed-forward approach—unless the initial data source is the same node as the ATM ingress. Figure 3 shows the feed-backward approach being used in subnet H. If the final switch on the path is congested (*), it doesn’t feed-forward any congestion indications on packet (U). Instead it sends a control cell (V) back to the router at the ATM ingress.
However, the backward feedback doesn’t reach the original data source directly because IP doesn’t support backward feedback (and subnet G is independent of subnet H). Instead, the router in the middle throttles down its sending rate but the original data sources don’t reduce their rates. The resulting rate mismatch causes the middle router’s buffer at layer 3 to back up until it becomes congested, which it signals forwards on later data packets at layer 3 (e.g. packet W). Note that the forward signal from the middle router is not triggered directly by the backward signal. Rather, it is triggered by congestion resulting from the middle router’s mismatched rate response to the backward signal.

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

4.4. Null Mode

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

A degenerate example is a point-to-point Ethernet link. Excess loading of the link merely causes the queue from the higher layer to back up, while the lower layer remains immune to congestion. Even a whole meshed subnetwork can be made immune to interior congestion by limiting ingress capacity and careful sizing of links, particularly if multi-path routing is used to ensure even worst-case patterns of load cannot congest any link.

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

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

The rest of this section is structured as follows:

- Section 5.1 addresses the most straightforward cases, where [RFC6040] can be applied directly to add ECN to tunnels that are effectively the same as IP-in-IP tunnels.
The subsequent sections give guidelines for adding ECN to a subnet technology that uses feed-forward-and-up mode like IP, but it is not so similar to IP that [RFC6040] rules can be applied directly. Specifically:

* Sections 5.2, 5.3 and 5.4 respectively address how to add ECN support to the wire protocol and to the encapsulators and decapsulators at the ingress and egress of the subnet.

* Section 5.5 deals with the special, but common, case of sequences of tunnels or subnets that all use the same technology

* Section 5.6 deals with the question of reframing when IP packets do not map 1:1 into lower layer frames.

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

A common pattern for many tunnelling protocols is to encapsulate an inner IP header with shim header(s) then an outer IP header. In many cases the shim header(s) always have to be tightly coupled to the outer IP header because they are not sufficient as outer headers in their own right. In such cases the shim header(s) and the outer IP header are always added (or removed) in the same operation. Therefore, in all such tightly coupled IP-in-IP tunnelling protocols, the rules in [RFC6040] for propagating the ECN field between the two IP headers SHOULD be applied directly.

Examples of tightly coupled IP-in-IP tunnelling protocols where [RFC6040] can be applied directly are:

- L2TP [RFC2661]
- GRE [RFC1701], [RFC2784]
- PPTP [RFC2637]
- GTP [GTPv1], [GTPv1-U], [GTPv2-C]
- VXLAN [RFC7348].

5.2. Wire Protocol Design: Indication of ECN Support

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

A lower layer (or subnet) congestion notification system:

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

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

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

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

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

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

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

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

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

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

5.3. Encapsulation Guidelines

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

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

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

* by the ingress explicitly checking that the egress propagates ECN (e.g. TRILL uses IS-IS to check path capabilities before using critical options [I-D.ietf-trill-rfc7180bis])

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

2. Egress Fails Capability Check: If the ingress cannot guarantee that the egress will propagate congestion notification, the ingress SHOULD disable ECN when it forwards the PDU at the lower layer. An example of how the ingress might disable ECN at the lower layer would be by setting the outer header of the PDU to identify it as a Not-ECN-PDU, assuming the subnet technology supports such a concept.

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

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

This guideline is intended to ensure that any bulk congestion monitoring of outer headers (e.g. by a network management node monitoring ECN in passing frames) is most meaningful. For instance, if an operator measures CE in 0.4% of passing outer headers, this information is only useful if the operator knows where the proportion of CE markings was last initialised to 0% (the Congestion Baseline). Such monitoring information will not be useful if some subnet ingress nodes reset all outer CE markings while others copy incoming CE markings into the outer.
Most information can be extracted if the Congestion Baseline is standardised at the node that is regulating the load (the Load Regulator--typically the data source). Then the operator can measure both congestion since the Load Regulator, and congestion since the subnet ingress. The latter might be measurable by subtracting the level of CE markings on inner headers from that on outer headers (see Appendix C of [RFC6040]).

5.4. Decapsulation Guidelines

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

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

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

   * If the outer header carries an explicit congestion marking, the packet SHOULD be dropped--the only indication of congestion that the L4 transport will understand.

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

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

3. In some lower layer protocols congestion may be signalled as a numerical level, such as in the control frames of quantised congestion notification [IEEE802.1Qau]. If such a multi-bit encoding encapsulates an ECN-capable IP data packet, a function will be needed to convert the quantised congestion level into the frequency of congestion markings in outgoing IP packets.
4. Congestion indications may be encoded by a severity level. For instance increasing levels of congestion might be encoded by numerically increasing indications, e.g. pre-congestion notification (PCN) can be encoded in each PDU at three severity levels in IP or MPLS [RFC6660].

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

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

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

5.5. Sequences of Similar Tunnels or Subnets

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

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

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

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

5.6. Reframing and Congestion Markings

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

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

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

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

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

The guidance in this section is applicable when IP packets:

- are encapsulated in Ethernet headers;
- 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].

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

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

- 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 (v4 or v6) when it may be encapsulated by more than one lower layer header, e.g. Ethernet MAC in MAC [IEEE802.1Qah].

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

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

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

3. It is sufficient to use the first IP header found in the stack; the egress of the relevant tunnel can propagate congestion.
7. Feed-Backward Mode: Guidelines for Adding Congestion Notification

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

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

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

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

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

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

This memo includes no request to IANA.

9. Security Considerations

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

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

- Congestion exposure (ConEx) for networks to audit that their congestion signals are not being suppressed by other networks or by receivers, and for networks to police that senders are responding sufficiently to the signals, irrespective of the transport protocol used [I-D.ietf-conex-abstract-mech].

- The ECN nonce [RFC3540] for a TCP sender to detect whether a network or the receiver is suppressing congestion signals.

- A test with the same goals as the ECN nonce, but without the need for the receiver to co-operate with the protocol [I-D.moncaster-tcpm-rcv-cheat].

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

10. Conclusions

Following the guidance in the document enables ECN support to be extended to numerous protocols that encapsulate IP (v4 & v6) in a consistent way, so that IP continues to fulfil its role as an end-to-end interoperability layer. This includes:
A wide range of tunnelling protocols with various forms of shim header between two IP headers;

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.

11. Acknowledgements

Thanks to Gorry Fairhurst for extensive reviews. Thanks also to the following reviewers: Ingemar Johansson and Piers O’Hanlon and Michael Welzl, who pointed out that lower layer congestion notification signals may have different semantics to those in IP.

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

12. Comments Solicited

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

13. References

13.1. Normative References


13.2. Informative References


[I-D.ietf-conex-abstract-mech]

[I-D.ietf-trill-rfc7180bis]

[I-D.moncaster-tcpm-rcv-cheat]

[IEEE802.1Qah]

(Access Controlled link within page)

[IEEE802.1Qau]

(Access Controlled link within page)

[ITU-T.I.371]


Appendix A. Outstanding Document Issues

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

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

2. Consider whether an IETF Standard Track doc will be needed to update the IP-in-IP protocols listed in Section 5.1—at least those that the IET

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

From ietf-01 to ietf-02:

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

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

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

From briscoe-03 to 04:

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

From briscoe-02 to 03:

* Scope section:
  + Added dependence on correct propagation of traffic class information
  + For the feed-backward mode, deemed multicast and anycast out of scope
* Ensured all guidelines referring to subnet technologies also refer to tunnels and vice versa by adding applicability
sentences at the start of sections 4.1, 4.2, 4.3, 4.4, 4.6 and 5.

* Added Security Considerations on ensuring congestion signal fields are classed as immutable and on using end-to-end congestion signal integrity technologies rather than hop-by-hop.

From briscoe-01 to 02:

* Added authors: JK & PT

* Added

  + Section 4.1 "IP-in-IP Tunnels with Tightly Coupled Shim Headers"

  + Section 4.5 "Sequences of Similar Tunnels or Subnets"

  + roadmap at the start of Section 4, given the subsections have become quite fragmented.

  + Section 9 "Conclusions"

* Clarified why transports are starting to be able to saturate interior links

* Under Section 1.1, addressed the question of alternative signal semantics and included multicast & anycast.

* Under Section 3.1, included a 3GPP example.

* Section 4.2. "Wire Protocol Design":

  + Altered guideline 2. to make it clear that it only applies to the immediate subnet egress, not later ones

  + Added a reminder that it is only necessary to check that ECN propagates at the egress, not whether interior nodes mark ECN

  + Added example of how QCN uses 802.1p to indicate support for QCN.

* Added references to Appendix C of RFC6040, about monitoring the amount of congestion signals introduced within a tunnel
* Appendix A: Added more issues to be addressed, including plan to produce a standards track update to IP-in-IP tunnel protocols.

* Updated acks and references

From briscoe-00 to 01:

* Intended status: BCP (was Informational) & updates 3819 added.

* Briefer Introduction: Introductory para justifying benefits of ECN. Moved all but a brief enumeration of modes of operation to their own new section (from both Intro & Scope). Introduced incr. deployment as most tricky part.

* Tightened & added to terminology section

* Structured with Modes of Operation, then Guidelines section for each mode.

* Tightened up guideline text to remove vagueness / passive voice / ambiguity and highlight main guidelines as numbered items.

* Added Outstanding Document Issues Appendix

* Updated references

Authors’ Addresses

Bob Briscoe
BT
B54/77, Adastral Park
Martlesham Heath
Ipswich  IP5 3RE
UK

Phone: +44 1473 645196
EMail: bob.briscoe@bt.com
URI: http://bobbriscoe.net/

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

EMail: john.kaippallimalil@huawei.com
Pat Thaler
Broadcom Corporation
5025 Keane Drive
Carmichael, CA  95608
USA

EMail: pthaler@broadcom.com
Abstract

This document describes a method of encapsulating network protocol packets within GRE and UDP headers. In this encapsulation, the source UDP port can be used as an entropy field for purposes of load balancing, while the protocol of the encapsulated packet in the GRE payload is identified by the GRE Protocol Type. Usage restrictions apply to GRE-in-UDP usage for traffic that is not congestion controlled and to UDP zero checksum usage with IPv6.

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 September 9, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

Load balancing, or more specifically statistical multiplexing of traffic using Equal Cost Multi-Path (ECMP) and/or Link Aggregation Groups (LAGs) in IP networks is a widely used technique for creating higher capacity networks out of lower capacity links. Most existing routers in IP networks are already capable of distributing IP traffic flows over ECMP paths and/or LAGs on the basis of a hash function performed on flow invariant fields in IP packet headers and their payload protocol headers. Specifically, when the IP payload is a User Datagram Protocol (UDP)[RFC768] or Transmission Control Protocol (TCP) [RFC793] packet, router hash functions frequently operate on the five-tuple of source IP address, destination IP address, source port, destination port, and protocol/next-header.

Several encapsulation techniques are commonly used in IP networks, such as Generic Routing Encapsulation (GRE) [RFC2784], MPLS [RFC4023] and L2TPv3 [RFC3931]. GRE is an increasingly popular encapsulation choice. Unfortunately, use of common GRE endpoints may reduce the entropy available for use in load balancing, especially in environments where the GRE Key field [RFC2890] is not readily available for use as entropy in forwarding decisions.

This document defines a generic GRE-in-UDP encapsulation for tunneling network protocol packets across an IP network. The GRE header provides payload protocol type as an EtherType in the protocol type field [RFC2784], and the UDP header provides additional entropy by way of its source port.

This encapsulation method requires no changes to the transit IP network. Hash functions in most existing IP routers may utilize and benefit from the use of a GRE-in-UDP tunnel without needing 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.

1.1. Applicability Statement

GRE encapsulation is widely used for many applications. For example, to redirect IP traffic to traverse a different path instead of the default path in an operator network, to tunnel private network traffic over a public network by use of public IP network addresses, to tunnel IPv6 traffic over an IPv4 network, and etc.

When using GRE-in-UDP encapsulation, encapsulated traffic will be treated as a UDP application, not as a GRE application, in an IP network.
network. Thus GRE-in-UDP tunnel needs to meet UDP application guidelines specified in [RFC5405bis], which can constrain GRE-in-UDP tunnel usage to certain applications and/or environments.

Here is the list of the UDP application guidelines in [RFC5405bis] and corresponding Sections to cover it in this document.

- Congestion Control: GRE-in-UDP does not have congestion control mechanism. The usage restrictions for traffic that is not congestion control is specified in Section 6.

- Message Size: Address in Section 4.1

- Reliability: not applicable to a GRE-in-UDP tunnel. GRE-in-UDP tunnel does not provide any reliable transport.

- Checksum: Address in Section 5.

- Middlebox Traversal: Section 5.2.1.

GRE-in-UDP encapsulation may be used to encapsulate already tunneled traffic, i.e. tunnel-in-tunnel. The tunneled traffic may use GRE-in-UDP or other tunnel encapsulation. In this case, GRE-in-UDP tunnel end points treat other tunnel endpoints as of the end hosts for the traffic and do not differentiate such end hosts from other end hosts.

2. Terminology

The terms defined in [RFC768][RFC2784] are used in this document.

2.1. Requirements Language

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

3. Encapsulation in UDP

GRE-in-UDP encapsulation format is shown as follows:
IPv4 Header:

+-----------------+-----------------+-----------------+-----------------+
<table>
<thead>
<tr>
<th>Version</th>
<th>IHL</th>
<th>Type of Service</th>
<th>Total Length</th>
</tr>
</thead>
</table>
+-----------------+-----------------+-----------------+-----------------+
| Identification | Flags | Fragment Offset |
+-----------------+-----------------+-----------------+-----------------+
| Time to Live | Protocol=17(UDP) | Header Checksum |
+-----------------+-----------------+-----------------+-----------------+
| Source IPv4 Address |
+-----------------+-----------------+-----------------+-----------------+
| Destination IPv4 Address |
+-----------------+-----------------+-----------------+-----------------+

UDP Header:

+-----------------+-----------------+-----------------+-----------------+
| Source Port = XXXX | Dest Port = TBD |
| 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
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 = XXXX | Dest Port = TBD |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 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. The TTL field in the IP header must be set to a value appropriate for delivery of the encapsulated packet to the peer of the encapsulation.

3.2. UDP Header

3.2.1. Source Port

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. IANA suggests this range to be 49152 to 65535, where the high order two bits of the port are set to one. This provides fourteen bits of entropy for the inner flow identifier. In the case that an encapsulator is unable to derive flow entropy from the payload header, it should set a randomly selected constant value for UDP source port to avoid payload packet flow reordering.

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.

How an encapsulator generates entropy from the payload is outside the scope of this document.

3.2.2. Destination Port

The destination port of the UDP header is set the GRE/UDP port (TBD) (see Section 8).

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 5.
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].

The GRE checksum MAY be enabled to protect the GRE header and payload. An encapsulator SHOULD NOT enable both the GRE checksum and UDP checksum simultaneously as this would be mostly redundant. Since the UDP checksum covers more of the packet including the GRE header and payload, the UDP checksum SHOULD have preference to using GRE checksum.

An implementation MAY use the GRE keyid to authenticate the encapsulator. In this model, a shared value is either configured or negotiated between an encapsulator and decapsulator. When a GRE-in-UDP packet is received with the keyid present, it is checked to see if it is valid for the source to have set for the tunnel packet was sent on. An implementation MAY enforce that a keyid be used for source authentication on selected tunnels. When a decapsulator determines a presented keyid is not valid for the source to send or the keyid is absent and is considered required for authenticating the encapsulator for a tunnel, the packet MUST be dropped.

4. Encapsulation Process Procedures

The GRE-in-UDP encapsulation allows encapsulated packets to be forwarded through "GRE-UDP tunnels". When performing GRE-in-UDP encapsulation by the encapsulator, the entropy value would be generated by the encapsulator and then be filled in the Source Port field of the UDP header. The Destination Port field is set to a value (TBD) allocated by IANA to indicate that the UDP tunnel payload is a GRE packet. The Protocol Type header field in GRE header is set to the EtherType value corresponding to the protocol of the encapsulated packet.
Intermediate routers, upon receiving these UDP encapsulated packets, could balance these packets based on the hash of the five-tuple of UDP packets.

Upon receiving these UDP encapsulated packets, the decapsulator would decapsulate them by removing the UDP and GRE headers and then process them accordingly.

Note: Each UDP tunnel is unidirectional, as GRE-in-UDP traffic is sent to the IANA-allocated UDP Destination Port, and in particular, is never sent back to any port used as a UDP Source Port (which serves solely as a source of entropy). This is at odds with a common middlebox (e.g., firewall) assumption that bidirectional traffic uses a common pair of UDP ports. As a result, arranging to pass bidirectional GRE-in-UDP traffic through middleboxes may require separate configuration for each direction of traffic.

GRE-in-UDP allows encapsulation of unicast, broadcast, or multicast traffic. Entropy may be generated from the header of encapsulated unicast or broadcast/multicast packets at an encapsulator. The mapping mechanism between the encapsulated multicast traffic and the multicast capability in the IP network is transparent and independent to the encapsulation and is otherwise outside the scope of this document.

To provide entropy for ECMP, GRE-in-UDP does not rely on GRE keep-alive. It is RECOMMENDED no use of 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 fragmentation, an encapsulator SHOULD perform fragmentation [GREMTU] on a packet before encapsulation and factor in both GRE and UDP header bytes in the effective Maximum Transmission Unit (MTU) size. Not performing the fragmentation will cause the packets exceeding network MTU size to be dropped or fragmented in the network. An encapsulator MUST use the same source UDP port for all packet fragments to ensure that the transit routers will forward the packet fragments on the same path. An operator should factor in the additional bytes of overhead when considering an MTU size for the payload to avoid the likelihood of fragmentation.

Fragmented packets MUST be reassembled at the decapsulator prior to being sent to a (payload) application. Packet fragmentation and reassembling process is outside the scope of the document.
4.2. Differentiated Services

To ensure that tunneled traffic gets the same treatment over the IP network, prior to the encapsulation process, an encapsulator should process the payload to get the proper parameters to fill into the IP header such as DiffServ [RFC2983]. Encapsulation end points that support ECN must use the method described in [RFC6040] for ECN marking propagation. This process is outside of the scope of this document.

5. UDP Checksum Handling

5.1. UDP Checksum with IPv4

For UDP in IPv4, the UDP checksum MUST be processed as specified in [RFC768] and [RFC1122] for both transmit and receive. An encapsulator MAY set the UDP checksum to zero for performance or implementation considerations. The IPv4 header includes a checksum which 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. Enabling or disabling the use of checksums is a deployment consideration that should take into account the risk and effects of packet corruption, and whether the packets in the network are protected by other, possibly stronger mechanisms such as the Ethernet CRC.

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.

5.2. 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, and so MUST used with the following exceptions:

a. Use of GRE-in-UDP in 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.

b. Use of GRE-in-UDP in networks under single administrative control (such as within a single operator’s network) where 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. Use of GRE-in-UDP for traffic delivery for 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.

For these exceptions, the UDP zero-checksum mode can be used. However the use of the UDP zero-checksum mode must meet the requirements specified in [RFC6935] and [RFC6936] as well at the additional requirements stated below.

These exceptions may also be extended to the use of GRE-in-UDP within a set of closely cooperating network administrations (such as network operators who have agreed to work together in order to jointly provide specific services).

As such, for IPv6, the UDP checksum for GRE-in-UDP MUST be used as specified in [RFC768] and [RFC2460] for tunnels that span multiple networks whose network administrations do not cooperate closely, even if each non-cooperating network administration independently satisfies one or more of the exceptions for UDP zero-checksum mode usage with GRE-in-UDP over IPv6.

The following additional requirements apply to implementation and use of UDP zero-checksum mode for GRE-in-UDP over IPv6:
a. Use of the UDP checksum with IPv6 MUST be the default configuration of all GRE-in-UDP implementations.

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

c. By default a decapsulator MUST disallow receipt of GRE-in-UDP packets with zero UDP checksums with IPv6. Zero checksums MAY selectively be enabled for certain source address. A 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.

d. An 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.

e. Any middlebox support for GRE-in-UDP with UDP zero-checksum mode for IPv6 MUST comply with requirements 1 and 8-10 in Section 5 of [RFC6936].

f. Measures SHOULD be taken to prevent IPv6 traffic with zero UDP checksums from "escaping" to the general Internet; see Section 6 for examples of such measures.

g. IPv6 traffic with zero UDP checksums MUST be actively monitored for errors by the network operator.

h. If a packet with a non-zero checksum is received, the checksum MUST be verified before accepting the packet. This is regardless of whether a 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. 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 GRE encapsulated packet corruption has not been a problem in practice.

Hence GRE-in-UDP is suitable for transmission over lower layers in the 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 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].

GRE does not accumulate incorrect state as a consequence of GRE header corruption. A corrupt GRE results in either packet discard or forwarding of the packet without accumulation of GRE state. GRE checksum MAY be used for protecting GRE header and payload. 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 inapplicable to GRE-in-UDP. Requirements 6 and 7 do not apply because GRE does not have a GRE-generic control feedback mechanism. Requirements 8-10 are middlebox requirements that do not apply to GRE-in-UDP tunnel endpoints, but see Section 5.2.1 for further middlebox discussion.

It is worth to mention 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 and without GRE checksums.

In summary, UDP zero-checksum mode for IPv6 is allowed to be used with GRE-in-UDP when one of the three exceptions specified above applies, provided that additional requirements stated above are complied with. Otherwise the UDP checksum MUST be used for IPv6 as specified in [RFC768] and [RFC2460]. Use of GRE checksum favors non-use of the UDP checksum.
5.2.1. Middlebox Considerations

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. 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. 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].

6. Congestion Considerations

Section 3.1.7 of [RFC5405bis] discussed the congestion implications of UDP tunnels. As discussed in [RFC5405bis], because other flows can share the path with one or more UDP tunnels, congestion control [RFC2914] needs to be considered.

A major motivation for GRE-in-UDP encapsulation is to tunnel a network protocol over IP network and improve the use of multipath (such as ECMP) in cases where traffic is to traverse routers which are able to hash on UDP Port and IP address. As such, in many cases this may reduce the occurrence of congestion and improve usage of available network capacity. However, it is also necessary to ensure that the network, including applications that use the network, responds appropriately in more difficult cases, such as when link or equipment failures have reduced the available capacity.

The impact of congestion must be considered both in terms of the effect on the rest of the network over which packets are sent in UDP tunnels, and in terms of the effect on the flows that are sent by UDP tunnels. The potential impact of congestion from a UDP tunnel depends upon what sort of traffic is carried over the tunnel, as well as the path of the tunnel.

GRE encapsulation is widely used to carry a wide range of network protocols and traffic. In many cases GRE encapsulation is used to carry IP traffic. IP traffic is generally assumed to be congestion controlled, and thus a tunnel carrying general IP traffic (as might be expected to be carried across the Internet) generally does not need additional congestion control mechanisms. As specified in RFC 5405:
"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."

For this reason, where GRE-in-UDP tunneling is used to carry IP traffic that is known to be congestion controlled, the UDP tunnels MAY be used within a single network or across multiple networks, with cooperating network operators. Internet IP traffic is generally assumed to be congestion-controlled.

However, GRE-in-UDP tunneling can be also used to carry traffic that is not necessarily congestion controlled. In such cases network operators may avoid congestion by careful provisioning of their networks, by rate limiting of user data traffic, and/or by using Traffic Engineering tools to monitor the network segments and dynamically steers traffic away from potentially congested links.

For this reason, where the GRE payload traffic is not congestion controlled, GRE-in-UDP tunnels MUST only be used within a single operator’s network that utilizes careful provisioning (e.g., rate limiting at the entries of the network while over-provisioning network capacity) to ensure against congestion, or within a limited number of networks whose operators closely cooperate in order to jointly provide this same careful provisioning.

As such, GRE-in-UDP MUST NOT be used over the general Internet, or over non-cooperating network operators, to carry traffic that is not congestion-controlled.

Measures SHOULD be taken to prevent non-congestion-controlled GRE-in-UDP traffic from "escaping" to the general Internet, e.g.:

- Physical or logical isolation of the links carrying GRE-in-UDP from the general Internet.
- Deployment of packet filters that block the UDP ports assigned for GRE-in-UDP.
- Imposition of restrictions on GRE-in-UDP traffic by software tools used to set up GRE-in-UDP tunnels between specific end systems (as might be used within a single data center).
7. Backward Compatibility

It is assumed that tunnel ingress routers must 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 router that is intended for use as a decapsulator does not support or enable GRE-in-UDP encapsulation described in this document, it will not be listening on destination port (TBD). In these cases, the router will conform to 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.

8. 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: TBD
Service Code: N/A
Known Unauthorized Uses: N/A
Assignment Notes: N/A

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>
9. Security Considerations

UDP and GRE encapsulation does not affect security for the payload protocol. When using GRE-in-UDP, Network Security in a network is mostly equivalent to that of a network using GRE.

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 usage 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 implementation that supports DTLS is expected to be able to establish DTLS sessions with multiple tunnel encapsulators, and likewise an GRE-in-UDP tunnel encapsulator implementation is expected to be able to establish DTLS sessions with multiple decapsulators (although different source and/or destination IP addresses may be involved - see Section 5.2 for discussion of one situation where use of different source IP addresses is important).

Use of ICMP for signaling of the GRE-in-UDP encapsulation capability adds a security concern. Upon receiving an ICMP message and before taking an action on it, the ingress MUST validate the IP address originating against tunnel egress address and MUST evaluate the packet header returned in the ICMP payload to ensure the source port is the one used for this tunnel. The mechanism for performing this validation is out of the scope of this document.

In an instance where the UDP source port is not set based on the flow invariant fields from the payload header, 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. How the source port randomization occurs is outside scope of this document.

Using one standardized value in UDP destination port for an encapsulation indication 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 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 header can cause a privacy and security concern for some applications that rely on the key field for traffic segregation. When GRE key field is used for privacy and security, ether UDP checksum or GRE checksum SHOULD be used for GRE-in-UDP with both IPv4 and IPv6, and in particular, when UDP zero-checksum mode is used, GRE checksum SHOULD be used.

10. Acknowledgements

Authors like to thank Vivek Kumar, Ron Bonica, Joe Touch, Ruediger Geib, Lar Edds, Lloyd, and many others for their review and valuable input on this draft.

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.

11. Contributors

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

David Black
EMC Corporation
176 South Street

Crabbe, et al.
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
12. References

12.1. Normative References


12.2. Informative References


13. Authors’ Addresses

Edward Crabbe
Email: edward.crabbe@gmail.com

Lucy Yong
Huawei Technologies, USA
Email: lucy.yong@huawei.com

Xiaohu Xu
Huawei Technologies,
Beijing, China
Email: xuxiaohu@huawei.com

Tom Herbert
Google
1600 Amphitheatre Parkway
Mountain View, CA
Email: tom@herbertland.com
Abstract

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

This document describes the protocol extensions required for the SCTP endpoints to help NATs provide similar features of NAPT in the single-point and multi-point traversal scenario.
Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction .............................................. 3
2. Conventions ............................................. 4
3. Terminology ............................................. 5
4. Motivation ................................................ 6
  4.1. SCTP NAT Traversal Scenarios ....................... 6
  4.1.1. Single Point Traversal ......................... 6
  4.1.2. Multi Point Traversal ......................... 7
4.2. Limitations of Classical NAPT for SCTP .......... 7
4.3. The SCTP Specific Variant of NAT ................ 8
5. Data Formats ............................................ 12
  5.1. Modified Chunks .................................... 12
  5.1.1. Extended ABORT Chunk ......................... 12
  5.1.2. Extended ERROR Chunk ......................... 12
  5.2. New Error Causes .................................. 13
  5.2.1. VTag and Port Number Collision Error Cause ... 13
  5.2.2. Missing State Error Cause .................... 13
  5.2.3. Port Number Collision Error Cause ............ 14
5.3. New Parameters ....................................... 14
  5.3.1. Disable Restart Parameter .................... 14
  5.3.2. VTags Parameter .................................. 15
6. Procedures ............................................... 16
  6.1. Overview ............................................ 16
  6.2. Association Setup Considerations ................. 17
  6.3. Handling of Internal Port Number and Verification Tag Collisions ................................ 17
  6.4. Handling of Internal Port Number Collisions ...... 18
  6.5. Handling of Missing State ......................... 19
  6.6. Handling of Fragmented SCTP Packets ............. 21
  6.7. Multi-Point Traversal Considerations ............ 21
7. Various Examples of NAT Traversals ................. 21
  7.1. Single-homed Client to Single-homed Server .... 21
1. Introduction

Stream Control Transmission Protocol [RFC4960] provides a reliable communications channel between two end-hosts in many ways similar to TCP [RFC0793]. With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT for TCP that allows multiple hosts to reside behind a NAT using private addresses (see [RFC6890]) and yet use only a single globally unique IPv4 address, even when two hosts (behind a NAT) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation (NAPT). To date, specialized code for SCTP has not yet been added to most NATs so that only true NAT is available. The end result of this is that only one SCTP capable host can be behind a NAT.

This document describes an SCTP specific variant NAT and specific packets and procedures to help NATs provide similar features of NAPT in the single-point and multi-point traversal scenario. An SCTP implementation supporting this extension will follow these procedures to assure that in both single-homed and multi-homed cases a NAT will maintain the proper state without needing to change port numbers.

The authors feel 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’s public IP address, much as TCP does today.
If a NAT does not need to change any data within an SCTP packet it will reduce the processing burden of NAT’ing SCTP by NOT needing to execute the CRC32c checksum required by SCTP.

Not having to touch the IP payload makes the processing of ICMP messages in NATs easier.

An SCTP-aware NAT will need to follow these procedures for generating appropriate SCTP packet formats.

When considering this feature it is possible to have multiple levels of support. At each level, the Internal Host, External Host and NAT may or may not support the features described in this document. The following table illustrates the results of the various combinations of support and if communications can occur between two endpoints.

<table>
<thead>
<tr>
<th>Internal Host</th>
<th>NAT</th>
<th>External Host</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td>Support</td>
<td>Support</td>
<td>Yes</td>
</tr>
<tr>
<td>Support</td>
<td>Support</td>
<td>No Support</td>
<td>Limited</td>
</tr>
<tr>
<td>Support</td>
<td>No Support</td>
<td>No Support</td>
<td>None</td>
</tr>
<tr>
<td>Support</td>
<td>No Support</td>
<td>No Support</td>
<td>Limited</td>
</tr>
<tr>
<td>No Support</td>
<td>Support</td>
<td>No Support</td>
<td>Limited</td>
</tr>
<tr>
<td>No Support</td>
<td>No Support</td>
<td>Support</td>
<td>None</td>
</tr>
<tr>
<td>No Support</td>
<td>No Support</td>
<td>No Support</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 1: Communication possibilities

From the table we can see that when a NAT does not support the extension no communication can occur. This is because for the most part of the current situation i.e. SCTP packets sent externally from behind a NAT are discarded by the NAT. In some cases, where the NAT supports the feature but one of the two external hosts does not support the feature, communication may occur but in a limited way. For example only one host may be able to have a connection when a collision case occurs.

2. Conventions

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

This document uses the following terms, which are depicted in Figure 1.

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

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

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

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

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

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

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

---

Figure 1: Basic network setup
4. Motivation

4.1. SCTP NAT Traversal Scenarios

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

4.1.1. Single Point Traversal

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

---

**Single NAT scenario**

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

---

**Serial NATs scenario**

In this single point traversal scenario, we must acknowledge that while one of the main benefits of SCTP multi-homing is redundant paths, the NAT function represents a single point of failure in the path of the SCTP multi-home association. However, the rest of the path may still benefit from path diversity provided by SCTP multi-homing.

The two SCTP endpoints in this case can be either single-homed or multi-homed. However, the important thing is that the NAT (or NATs) in this case sees all the packets of the SCTP association.
4.1.2. Multi Point Traversal

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

```
+----------------+                  +----------------+
| SCTP end point A |                  | SCTP end point B |
|                 |                  |                 |
| +----------------+                  +----------------+ |
| |                 |                  |                 | |
| | Internet        |                  |                 | |
| |                  |                  |                 | |
| +----------------+                  +----------------+ |
| | NAT B           |                  |                 | |
| |                 |                  |                 | |
| +----------------+                  +----------------+ |
| |                 |                  |                 | |
| | NAT A           |                  |                 | |
| |                 |                  |                 | |
| +----------------+                  +----------------+ |
| |                 |                  |                 | |
| +----------------+                  +----------------+ |
| |                 |                  |                 | |
| +----------------+                  +----------------+ |
| |                 |                  |                 | |
| +----------------+                  +----------------+ |
| |                 |                  |                 | |
| +----------------+                  +----------------+ |
```

Parallel NATs scenario

This case does NOT apply to a single-homed SCTP association (i.e., BOTH endpoints in the association use only one IP address). The advantage here is that the existence of multiple NAT traversal points can preserve the path diversity of a multi-homed association for the entire path. This in turn can improve the robustness of the communication.

4.2. Limitations of Classical NAPT for SCTP

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

An SCTP endpoint may have multiple addresses but only has a single port number. To make multipoint traversal work, all the NATs involved must recognize the packets they see as belonging to the same SCTP association and perform port number translation in a consistent way. One possible way of doing this is to use pre-defined table of ports and addresses configured within each NAT. Other mechanisms could make use of NAT to NAT communication. Such mechanisms are considered by the authors not to be deployable on a wide scale base and thus not a recommended solution. Therefore the SCTP variant of NAT has been developed.
4.3. The SCTP Specific Variant of NAT

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

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

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

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

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

The processing of outgoing SCTP packets containing an INIT-chunk is described in the following figure. The scenario shown is valid for all message flows in this section.
INIT[Initiate-Tag]
Priv-Addr:Int-Port -------> Ext-Addr:Ext-Port
Ext-VTag=0
Create(Initiate-Tag, Int-Port, Priv-Addr, 0)
Returns(NAT-State control block)
Translate To:

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

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

It is also possible that a connection to External-Address and External-Port exists without an Internal-VTag conflict but the External-Address does not support the DISABLE_RESTART feature (noted in the NAT control block when the prior connection was established). In such a case the INIT SHOULD be dropped by the NAT and an ABORT SHOULD be sent back to the SCTP host with the M-Bit set and an appropriate error cause (see Section 5.1.1 for the format).

The processing of outgoing SCTP packets containing no INIT-chunk is described in the following figure.
The processing of incoming SCTP packets containing INIT-ACK chunks is described in the following figure. The Lookup() function getting as input the Internal-VTag, Internal-Port, External-VTag (=0), External-Port, and External-Address, returns the corresponding entry of the NAT table and updates the External-VTag by substituting it with the value of the Initiate-Tag of the INIT-ACK chunk. The wildcard character signifies that the parameter’s value is not considered in the Lookup() function or changed in the Update() function, respectively.

```
                  +--------+          +-----+           /        \
[Host A] <------> [NAT] <------> [Internet] <------> [Host B]
                  \--------+          \-----+           \        /\---/
Priv-Addr:Int-Port ------> Ext-Addr:Ext-Port
Ext-VTag

Translate To:

          +--------+          +-----+           /        \
[Host A] <------> [NAT] <------> [Internet] <------> [Host B]
          \--------+          \-----+           \        /---\
Pub-Addr:Int-Port ------> Ext-Addr:Ext-Port
Ext-VTag

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

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

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

INIT-ACK[Initiate-Tag]
Priv-Addr:Int-Port ------> Ext-Addr:Ext-Port
Int-VTag
```
In the case Lookup fails, the SCTP packet is dropped. The Update routine inserts the External-VTag (the Initiate-Tag of the INIT-ACK chunk) in the NAT state control block.

The processing of incoming SCTP packets containing an ABORT or SHUTDOWN-COMPLETE chunk with the T-Bit set is described in the following figure.

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

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

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

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

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

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

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

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

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

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

Priv-Addr:Int-Port <------ Ext-Addr:Ext-Port
Int-VTag
For an incoming packet containing an INIT-chunk a table lookup is made only based on the addresses and port numbers. If an entry with an External-VTag of zero is found, it is considered a match and the External-VTag is updated.

This allows the handling of INIT-collision through NAT.

5. Data Formats

5.1. Modified Chunks

This section presents existing chunks defined in [RFC4960] that are modified by this document.

5.1.1. Extended ABORT Chunk

The ABORT chunk is extended to add the new ‘M-bit’. The M-bit indicates to the receiver of the ABORT chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box.

5.1.2. Extended ERROR Chunk

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

<table>
<thead>
<tr>
<th>Cause Code = 0x00B0</th>
<th>Cause Length = Variable</th>
</tr>
</thead>
</table>
\[ Chunk \]

Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the VTag and Port Number Collision Error Cause. The suggested value of this field for IANA is 0x00B0.

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.

5.2.2. Missing State Error Cause

<table>
<thead>
<tr>
<th>Cause Code = 0x00B1</th>
<th>Cause Length = Variable</th>
</tr>
</thead>
</table>
\[ Incoming Packet \]

Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the Missing State Error Cause. The suggested value of this field for IANA is 0x00B1.

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.

Incoming 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.

5.2.3. Port Number Collision Error Cause

```
+-----------------+-----------------+-----------------+
| Cause Code = 0x00B2 | Cause Length = | Cause Specific   |
|                  | Variable        | Information     |
+-----------------+-----------------+-----------------+
```

Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the Port Number Collision Error Cause. The suggested value of this field for IANA is 0x00B2.

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.

5.3. New Parameters

This section defines new parameters and their valid appearance defined by this document.

5.3.1. Disable Restart Parameter

This parameter is used to indicate that the RESTART procedure is requested to be disabled. Both endpoints of an association MUST include this parameter in the INIT chunk and INIT-ACK chunk when
establishing an association and MUST include it in the ASCONF chunk when adding an address to successfully disable the restart procedure.

Parameter Type: 2 bytes (unsigned integer)
This field holds the IANA defined parameter type for the Disable Restart Parameter. The suggested value of this field for IANA is 0xC007.

Parameter Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the parameter. The value MUST be 4.

This 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 recover from state loss.

Parameter Type: 2 bytes (unsigned integer)
This field holds the IANA defined parameter type for the VTags Parameter. The suggested value of this field for IANA is 0xC008.

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 ASCONF can use this same value in the ASCONF-ACK to find which request the response is for. Note that the receiver MUST NOT change this 32-bit value.

Internal Verification Tag: 4 bytes (unsigned integer)  The Verification Tag that the internal host has chosen for its communication. The Verification Tag is a unique 32-bit tag that must accompany any incoming SCTP packet for this association to the Private-Address.

External Verification Tag: 4 bytes (unsigned integer)  The Verification Tag that the host holding the External-Address has chosen for its communication. The VTag is a unique 32-bit tag that must accompany any incoming SCTP packet for this association to the External-Address.

This parameter MAY appear in ASCONF chunks and MUST NOT appear in any other chunk.

6. Procedures

6.1. Overview

When an SCTP endpoint is behind an SCTP-aware NAT a number of problems may arise as it tries to communicate with its peer:

o More than one host behind a NAT may pick the same VTag and source port when talking to the same peer server. This creates a situation where the NAT will not be able to tell the two associations apart. This situation is discussed in Section 6.3.

o When an SCTP endpoint is a server communicating with multiple peers and the peers are behind the same NAT, then the two endpoints cannot be distinguished by the server. This case is discussed in Section 6.4.

o A restart of a NAT during a conversation could cause a loss of its state. This problem and its solution is discussed in Section 6.5.

o An SCTP endpoint may be behind two NATs providing redundancy. The method to set up this scenario is discussed in Section 6.7.

Each of these mechanisms requires additional chunks and parameters, defined in this document, and possibly modified handling procedures from those specified in [RFC4960] fdafdafdafdafdafdasf.

6.2. Association Setup Considerations

Every association MUST initially be set up single-homed. There MUST NOT be any IPv4 Address parameter, IPv6 Address parameter, or Supported Address Types parameter in the INIT-chunk. The INIT-ACK chunk MUST NOT contain any IPv4 Address parameter or IPv6 Address parameter.

If the association should finally be multi-homed, the procedure in Section 6.7 MUST be used.

The INIT and INIT-ACK chunk SHOULD contain the Disable Restart parameter defined in Section 5.3.1.

6.3. Handling of Internal Port Number and Verification Tag Collisions

Consider the case where two hosts in the Private-Address space want to set up an SCTP association with the same server running on the same host in the Internet. This means that the External-Port and the External-Address are the same. If they both choose the same Internal-Port and Internal-VTag, the NAT box cannot distinguish between incoming packets anymore. But this is very unlikely. The Internal-VTags are chosen at random and if the Internal-Ports are also chosen from the ephemeral port range at random this gives a 46-bit random number which has to match. In the TCP like NAPT case the NAT box can control the 16-bit Natted Port and therefore avoid collisions deterministically.

The same can happen when an INIT-ACK chunk or an ASCONF chunk is processed by the NAT.

However, in this unlikely event the NAT box MUST send an ABORT chunk with the M-bit set if the collision is triggered by an INIT or INIT-ACK chunk or send an ERROR chunk with the M-bit set if the collision is triggered by an ASCONF chunk. 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. In the other two cases, the source and destination address and port numbers MUST be swapped.

The sender of the packet containing the INIT chunk or the receiver of the INIT-ACK chunk, upon reception of an ABORT chunk with M-bit set, SHOULD reinitiate the association setup procedure after choosing a new initiate tag. These procedures SHOULD be followed only if the appropriate error cause code for colliding NAT table state is
included AND the association is in the COOKIE-WAIT state (i.e. it is awaiting an INIT-ACK). If the endpoint is in any other state an SCTP endpoint SHOULD NOT respond.

The sender of the ASCONF chunk, upon reception of an ERROR chunk with M-bit set, MUST stop adding the path to the association.

The sender of the ERROR or ABORT chunk MUST include the error cause with cause code ‘VTag and Port Number Collision’ (see Section 5.2.1).

6.4. 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 Private-Address space will want to set up an SCTP association with the same server running on the same host in the Internet. For the NAT appropriate tracking may be performed by assuring that the VTags are unique between the two hosts. But for the external SCTP server on the internet this means that the External-Port and the External-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 MUST do the following:

- Include a Disable Restart parameter in the INIT-ACK to inform the client that it will support the feature.
- 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) differentiated only by the VTags.

The NAT, when processing the INIT-ACK, should note in its internal table that the association supports the Disable Restart extension. This note is used when establishing future associations (i.e. when processing an INIT from an internal host) to decide if the connection should be allowed. The NAT MUST do the following when processing an INIT:

- If the INIT is destined to an external address and port for which the NAT has no outbound connection, allow the INIT creating an internal mapping table.
If the INIT matches the external address and port of an already existing connection, validate that the external server supports the Disable Restart feature, if it does allow the INIT to be forwarded.

If the external server does not support the Disable Restart extension the NAT MUST send an ABORT with the M-bit set.

The ‘Port Number Collision’ error cause (see Section 5.2.3) MUST be included in the ABORT chunk.

If the collision is triggered by an ASCONF chunk, a packet containing an ERROR chunk with the ‘Port Number Collision’ error cause MUST be sent back.

6.5. Handling of Missing State

If the NAT box receives a packet from the internal network for which the lookup procedure does not find an entry in the NAT table, a packet containing an ERROR chunk is sent back with the M-bit set. The source address of the packet containing the ERROR chunk MUST be the destination address of the incoming SCTP packet. The verification tag is reflected and the T-bit is set. Please note that such a packet containing an ERROR chunk SHOULD NOT be sent if the received packet contains an ABORT, SHUTDOWN-COMPLETE or INIT-ACK chunk. An ERROR chunk MUST NOT be sent if the received packet contains an ERROR chunk with the M-bit set.

When sending the ERROR chunk, the new error cause Missing state (see Section 5.2.2) MUST be included and the new M-bit of the ERROR chunk MUST be set (see Section 5.1.2).

Upon reception of this ERROR chunk by an SCTP endpoint the receiver SHOULD take the following actions:

- Validate that the verification tag is reflected by looking at the VTag that would have been included in the outgoing packet.

- Validate that the peer of the SCTP association supports the dynamic address extension, if it does not discard the incoming ERROR chunk.

- Generate a new ASCONF chunk containing the VTtags parameter (see Section 5.3.2) and the Disable Restart parameter if the association is using the disabled restart feature. By processing this packet the NAT can recover the appropriate state. The procedures for generating an ASCONF chunk can be found in [RFC5061].
If the NAT box receives a packet for which it has no NAT table entry and the packet contains an ASCONF chunk with the VTags parameter, the NAT box MUST update its NAT table according to the verification tags in the VTags parameter and the optional Disable Restart parameter.

The peer SCTP endpoint receiving such an ASCONF chunk SHOULD either add the address and respond with an acknowledgment, if the address is new to the association (following all procedures defined in [RFC5061]). Or, if the address is already part of the association, the SCTP endpoint MUST NOT respond with an error, but instead should respond with an ASCONF-ACK chunk acknowledging the address but take no action (since the address is already in the association).

Note that it is possible that upon receiving an ASCONF chunk containing the VTags parameter the NAT will realize that it has an ‘Internal Port Number and Verification Tag collision’. In such a case the NAT MUST send 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 an ERROR 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 required. The endpoint MUST do the following:

- Validate that the verification tag is reflected by looking at the VTag that would have been included in the outgoing packet.
- Validate that the peer of the SCTP association supports the dynamic address extension, if it does not discard the incoming ERROR chunk.
- If the association is attempting to add an address (i.e. following the procedures in Section 6.7) 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 has a VTag collision and the association cannot easily create a new VTag (as it would if the error occurred when sending an INIT).
- If the endpoint has no other path, i.e. the procedure was executed due to missing a state in the NAT, then the endpoint MUST abort the association. This would occur only if the local NAT 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 looses its state).
6.6. Handling of Fragmented SCTP Packets

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

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

6.7. Multi-Point Traversal Considerations

If a multi-homed SCTP endpoint behind a NAT connects to a peer, it SHOULD first set up the association single-homed with only one address causing the first NAT to populate its state. Then it SHOULD add each IP address using ASCONF chunks sent via their respective NATs. The address to add is the wildcard address and the lookup address SHOULD also contain the VTags parameter and optionally the Disable Restart parameter as illustrated above.

7. Various Examples of NAT Traversals

7.1. Single-homed Client to Single-homed Server

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

```
+--------+          +-----+           /        \
| Host A | <------> | NAT | <------> | Internet | <------> | Host B |
\        /          \         /          +--------+
\--/---|          |--/---|          \--/---/\---/---/
      \--------+--------+-----------+----------+--------+
       NAT     Int    VTag |     Int    Port |  Priv    Addr |     Ext    VTag |   Ext    Port |
       +---------------------+---------------------+
           INIT[Initiate-Tag = 1234]

10.0.0.1:1 -----> 100.0.0.1:2
     Ext-VTag = 0
```

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

<table>
<thead>
<tr>
<th>NAT</th>
<th>Int</th>
<th>Priv</th>
<th>Ext</th>
<th>Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VTag</td>
<td>Port</td>
<td>VTag</td>
<td>Port</td>
</tr>
<tr>
<td>1234</td>
<td>1</td>
<td>10.0.0.1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

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

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

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

NAT updates entry:

<table>
<thead>
<tr>
<th>NAT</th>
<th>Int</th>
<th>Priv</th>
<th>Ext</th>
<th>Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VTag</td>
<td>Port</td>
<td>VTag</td>
<td>Port</td>
</tr>
<tr>
<td>1234</td>
<td>1</td>
<td>10.0.0.1</td>
<td>5678</td>
<td>2</td>
</tr>
</tbody>
</table>

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

The handshake finishes with a COOKIE-ECHO acknowledged by a COOKIE-ACK.
7.2. Single-homed Client to Multi-homed Server

The internal client is single-homed whereas the external server is multi-homed. The client (Host A) sends an INIT like in the single-homed case.
INIT[Initiate-Tag = 1234]
10.0.0.1:1 ---> 100.0.0.1:2
Ext-VTag = 0

NAT creates entry:

<table>
<thead>
<tr>
<th>VTag</th>
<th>Port</th>
<th>Addr</th>
<th>VTag</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>1</td>
<td>10.0.0.1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

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

The server (Host B) includes its two addresses in the INIT-ACK chunk, which results in two NAT entries.
NAT does need to change the table for second address:

```
+---------+--------+-----------+----------+--------+
| NAT    |  Int    |  Int   |    Priv   |   Ext    |   Ext  |
|  VTag   |  Port  |    Addr   |   VTag   |   Port |
+---------+--------+-----------+----------+--------+
|  1234   |    1   |  10.0.0.1 |    5678  |    2   |
```  

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

The handshake finishes with a COOKIE-ECHO acknowledged by a COOKIE-ACK.
### 7.3. Multihomed Client and Server

The client (Host A) sends an INIT to the server (Host B), but does not include the second address.
INIT[Initiate-Tag = 1234]
10.0.0.1:1 --------> 100.0.0.1:2
Ext-VTag = 0

NAT 1 creates entry:

<table>
<thead>
<tr>
<th>VTag</th>
<th>Port</th>
<th>Priv Addr</th>
<th>Ext VTag</th>
<th>Ext Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>1</td>
<td>10.0.0.1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

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

Host B includes its second address in the INIT-ACK, which results in two NAT entries in NAT 1.
INIT-ACK[Initiate-Tag = 5678, IP-Addr = 100.1.0.1]
101.0.0.1:1 <------------------------- 100.0.0.1:2
Int-VTag = 1234

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

<table>
<thead>
<tr>
<th>Int VTag</th>
<th>Int Port</th>
<th>Priv Addr</th>
<th>Ext VTag</th>
<th>Ext Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>1</td>
<td>10.0.0.1</td>
<td>5678</td>
<td>2</td>
</tr>
</tbody>
</table>

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

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

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

NAT 2 creates complete entry:
7.4. NAT Loses Its State

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

NAT 2

<table>
<thead>
<tr>
<th>Int VTag</th>
<th>Int Port</th>
<th>Priv Addr</th>
<th>Ext VTag</th>
<th>Ext Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>1</td>
<td>10.1.0.1</td>
<td>5678</td>
<td>2</td>
</tr>
</tbody>
</table>

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

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

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

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

The NAT box cannot find entry for the association. It sends ERROR message with the M-Bit set and the cause "NAT state missing".
ERROR [M-Bit, NAT state missing]
10.0.0.1:1 <---------- 100.0.0.1:2
Ext-VTag = 5678

On reception of the ERROR message, Host A sends an ASCONF chunk
indicating that the former information has to be deleted and the
source address of the actual packet added.

ASCONF [ADD-IP,DELETE-IP,Int-VTag=1234, Ext-VTag = 5678]
10.0.0.1:1 ----------> 100.1.0.1:2
Ext-VTag = 5678

NAT
<table>
<thead>
<tr>
<th>Int VTag</th>
<th>Int Port</th>
<th>Priv Addr</th>
<th>Ext VTag</th>
<th>Ext Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>1</td>
<td>10.0.0.1</td>
<td>5678</td>
<td>2</td>
</tr>
</tbody>
</table>

Host B adds the new source address and deletes all former entries.
7.5. Peer-to-Peer Communication

If two hosts are behind NATs, they have to get knowledge of the peer’s public address. This can be achieved with a so-called rendezvous server. Afterwards the destination addresses are public, and the association is set up with the help of the INIT collision. The NAT boxes create their entries according to their internal peer’s point of view. Therefore, NAT A’s Internal-VTag and Internal-Port are NAT B’s External-VTag and External-Port, respectively. The naming of the verification tag in the packet flow is done from the sending peer’s point of view.
INIT[Initiate-Tag = 1234]
10.0.0.1:1 --> 100.0.0.1:2
Ext-VTag = 0

NAT A creates entry:

<table>
<thead>
<tr>
<th>Int VTag</th>
<th>Int Port</th>
<th>Priv Addr</th>
<th>Ext VTag</th>
<th>Ext Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>1</td>
<td>10.0.0.1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

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

NAT B processes INIT, but cannot find an entry. The SCTP packet is silently discarded and leaves the NAT table of NAT B unchanged.
Now Host B sends INIT, which is processed by NAT B. Its parameters are used to create an entry.

```
Init[Initiate-Tag = 5678]  
101.0.0.1:1 <-- 10.1.0.1:2  
Ext-VTag = 0
```

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

```
Init[Initiate-Tag = 5678]  
101.0.0.1:1  <--------------- 100.0.0.1:2  
Ext-VTag = 0
```
VTag != Int-VTag, but Ext-VTag == 0, find entry.

<table>
<thead>
<tr>
<th>VTag</th>
<th>Port</th>
<th>Addr</th>
<th>VTag</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>1</td>
<td>10.0.0.1</td>
<td>5678</td>
<td>2</td>
</tr>
</tbody>
</table>

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

Host A send INIT-ACK, which can pass through NAT B:
INIT-ACK[Initiate-Tag = 1234]
10.0.0.1:1 -->; 100.0.0.1:2
Ext-VTag = 5678

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

NAT B updates entry:

initiate-Tag = 1234
101.0.0.1:1 --> 10.1.0.1:2
Ext-VTag = 5678

The lookup for COOKIE-ECHO and COOKIE-ACK is successful.
8. 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.
8.1. 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.

```c
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 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.

9. 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 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 suggested changes are described below.

9.1. 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 suggested 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:
As defined in [RFC6096] one chunk flag has to be assigned by IANA for the ABORT chunk. The suggested value of the M bit is 0x02.

This requires an update of the "ABORT Chunk Flags" registry for SCTP:

```
<table>
<thead>
<tr>
<th>Chunk Flag Value</th>
<th>Chunk Flag Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>T bit</td>
<td>[RFC4960]</td>
</tr>
<tr>
<td>0x02</td>
<td>M bit</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>0x04</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0x40</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0x80</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>
```

9.2. Three New Error Causes

Three error causes have to be assigned by IANA. It is suggested to use the values given below.

This requires three additional lines in the "Error Cause Codes" registry for SCTP:
9.3. Two New Chunk Parameter Types

Two chunk parameter types have to be assigned by IANA. It is suggested to use the values given below. IANA should assign these values from the pool of parameters with the upper two bits set to ‘11’.

This requires two additional lines in the "Chunk Parameter Types" registry for SCTP:

<table>
<thead>
<tr>
<th>ID Value</th>
<th>Chunk Parameter Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>49159</td>
<td>Disable Restart (0xC007)</td>
<td>[RFCxxxx]</td>
</tr>
<tr>
<td>49160</td>
<td>VTags (0xC008)</td>
<td>[RFCxxxx]</td>
</tr>
</tbody>
</table>

10. Security Considerations

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

For SCTP end-points, this document does not add any additional security considerations to the ones given in [RFC4960], [RFC4895], and [RFC5061].

11. Acknowledgments

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

12.1. Normative References


12.2. Informative References


Authors' Addresses

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
US

Email: randall@lakerest.net
Recommendations on Using Assigned Transport Port Numbers
draft-ietf-tsvwg-port-use-10.txt

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on September 25, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

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

1. Introduction

This document provides information and advice to application and service designers on the use of assigned transport port numbers. It provides a detailed historical background of the evolution of transport port numbers and their multiple meanings. It also provides specific recommendations to designers on how to use assigned port numbers. Note that this document provides information to potential port number applicants that complements the IANA process described in BCP165 [RFC6335], but it does not change any of the port number
assignment procedures described therein. This document is intended to address concerns typically raised during Expert Review of assigned port number applications, but it is not intended to bind those reviews. RFC 6335 also describes the interaction between port experts and port requests in IETF consensus document. Authors of IETF consensus documents should nevertheless follow the advice in this document and can expect comment on their port requests from the port experts during IETF last call or at other times when review is explicitly sought.

2. Conventions used in this document

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

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

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

3. History

The term ‘port’ was first used in [RFC33] to indicate a simplex communication path from an individual process and originally applied to only the Network Control Program (NCP) connection-oriented protocol. At a meeting described in [RFC37], an idea was presented to decouple connections between processes and links that they use as paths, and thus to include numeric source and destination socket identifiers in packets. [RFC38] provides further detail, describing how processes might have more than one of these paths and that more than one path may be active at a time. As a result, there was the need to add a process identifier to the header of each message so that incoming messages could be demultiplexed to the appropriate process. [RFC38] further suggested that 32 bit numbers would be used for these identifiers. [RFC48] discusses the current notion of listening on a specific port number, but does not discuss the issue of port number determination. [RFC61] notes that the challenge of knowing the appropriate port numbers is "left to the processes" in general, but introduces the concept of a "well-known" port number for common services.
[RFC76] proposed a "telephone book" by which an index would allow port numbers to be used by name, but still assumed that both source and destination port numbers are fixed by such a system. [RFC333] proposed that a port number pair, rather than an individual port number, would be used on both sides of the connection for demultiplexing messages. This is the final view in [RFC793] (and its predecessors, including [IEN112]), and brings us to their current meaning. [RFC739] introduced the notion of generic reserved port numbers for groups of protocols, such as "any private RJE server" [RFC739]. Although the overall range of such port numbers was (and remains) 16 bits, only the first 256 (high 8 bits cleared) in the range were considered assigned.

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

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-63</td>
<td>0-77</td>
</tr>
<tr>
<td>64-127</td>
<td>100-177</td>
</tr>
<tr>
<td>128-223</td>
<td>200-337</td>
</tr>
<tr>
<td>224-255</td>
<td>340-377</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1023</td>
<td>0x0000-0x03FF</td>
<td>System (also Well-Known)</td>
</tr>
<tr>
<td>1024-49151</td>
<td>0x0400-0xBFFF</td>
<td>User (also Registered)</td>
</tr>
<tr>
<td>49152-65535</td>
<td>0xC000-0xFFFF</td>
<td>Dynamic (also Private)</td>
</tr>
</tbody>
</table>

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:

- Demultiplexing transport endpoint associations within an end host
- 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 can also simplify firewall management, so that a single, fixed firewall configuration can either permit or deny a service.

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

Assigned port numbers are useful for configuring firewalls and other port-based systems for access control. Ultimately, these 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. However, assigned port numbers often are important in helping configure firewalls.

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].
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:

- Use of a discovery service, either a shared service (mDNS), or a discovery service for a given system [RFC6762] [RFC6763].

- 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 it is still important to have assigned port numbers, largely to traverse either NATs or firewalls. Although
NAT traversal protocols supporting automatic configuration have been proposed and developed (e.g., STUN [RFC5389], TURN [RFC5766], and ICE [RFC5245]), application and service designers cannot yet rely on their presence.

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.
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


10.2. Informative References


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

Abstract

The User Datagram Protocol (UDP) provides a minimal message-passing transport that has no inherent congestion control mechanisms. Because congestion control is critical to the stable operation of the Internet, applications and other protocols that choose to use UDP as an Internet transport must employ mechanisms to prevent congestion collapse and to establish some degree of fairness with concurrent traffic. They may also need to implement additional mechanisms, depending on how they use UDP.

This document provides guidelines on the use of UDP for the designers of applications, tunnels and other protocols that use UDP. Congestion control guidelines are a primary focus, but the document also provides guidance on other topics, including message sizes, reliability, checksums, and middlebox traversal.

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

Status of This Memo

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

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

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

This Internet-Draft will expire on August 23, 2015.
1. Introduction

The User Datagram Protocol (UDP) [RFC0768] provides a minimal, unreliable, best-effort, message-passing transport to applications and other protocols (such as tunnels) that desire to operate over UDP (both simply called "applications" in the remainder of this document). Compared to other transport protocols, UDP and its UDP-
Lite variant [RFC3828] are unique in that they do not establish end-to-end connections between communicating end systems. UDP communication consequently does not incur connection establishment and tear-down overheads, and there is minimal associated end system state. Because of these characteristics, UDP can offer a very efficient communication transport to some applications.

A second unique characteristic of UDP is that it provides no inherent congestion control mechanisms. On many platforms, applications can send UDP datagrams at the line rate of the link interface, which is often much greater than the available path capacity, and doing so contributes to congestion along the path. [RFC2914] describes the best current practice for congestion control in the Internet. It identifies two major reasons why congestion control mechanisms are critical for the stable operation of the Internet:

1. The prevention of congestion collapse, i.e., a state where an increase in network load results in a decrease in useful work done by the network.

2. The establishment of a degree of fairness, i.e., allowing multiple flows to share the capacity of a path reasonably equitably.

Because UDP itself provides no congestion control mechanisms, it is up to the applications that use UDP for Internet communication to employ suitable mechanisms to prevent congestion collapse and establish a degree of fairness. [RFC2309] discusses the dangers of congestion-unresponsive flows and states that "all UDP-based streaming applications should incorporate effective congestion avoidance mechanisms". This is an important requirement, even for applications that do not use UDP for streaming. In addition, congestion-controlled transmission is of benefit to an application itself, because it can reduce self-induced packet loss, minimize retransmissions, and hence reduce delays. Congestion control is essential even at relatively slow transmission rates. For example, an application that generates five 1500-byte UDP datagrams in one second can already exceed the capacity of a 56 Kb/s path. For applications that can operate at higher, potentially unbounded data rates, congestion control becomes vital to prevent congestion collapse and establish some degree of fairness. Section 3 describes a number of simple guidelines for the designers of such applications.

A UDP datagram is carried in a single IP packet and is hence limited to a maximum payload of 65,507 bytes for IPv4 and 65,527 bytes for IPv6. The transmission of large IP packets usually requires IP fragmentation. Fragmentation decreases communication reliability and efficiency and should be avoided. IPv6 allows the option of
transmitting large packets ("jumbograms") without fragmentation when all link layers along the path support this [RFC2675]. Some of the guidelines in Section 3 describe how applications should determine appropriate message sizes. Other sections of this document provide guidance on reliability, checksums, and middlebox traversal.

This document provides guidelines and recommendations. Although most UDP applications are expected to follow these guidelines, there do exist valid reasons why a specific application may decide not to follow a given guideline. In such cases, it is RECOMMENDED that application designers cite the respective section(s) of this document in the technical specification of their application or protocol and explain their rationale for their design choice.

[RFC5405] was scoped to provide guidelines for unicast applications only, whereas this document also provides guidelines for UDP flows that use IP anycast, multicast and broadcast, and applications that use UDP tunnels to support IP flows.

Finally, although this document specifically refers to applications that use UDP, the spirit of some of its guidelines also applies to other message-passing applications and protocols (specifically on the topics of congestion control, message sizes, and reliability). Examples include signaling or control applications that choose to run directly over IP by registering their own IP protocol number with IANA. This document may provide useful background reading to the designers of such applications and protocols.

2. Terminology

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

3. UDP Usage Guidelines

Internet paths can have widely varying characteristics, including transmission delays, available bandwidths, congestion levels, reordering probabilities, supported message sizes, or loss rates. Furthermore, the same Internet path can have very different conditions over time. Consequently, applications that may be used on the Internet MUST NOT make assumptions about specific path characteristics. They MUST instead use mechanisms that let them operate safely under very different path conditions. Typically, this requires conservatively probing the current conditions of the Internet path they communicate over to establish a transmission...
behavior that it can sustain and that is reasonably fair to other traffic sharing the path.

These mechanisms are difficult to implement correctly. For most applications, the use of one of the existing IETF transport protocols is the simplest method of acquiring the required mechanisms. Consequently, the RECOMMENDED alternative to the UDP usage described in the remainder of this section is the use of an IETF transport protocol such as TCP [RFC0793], Stream Control Transmission Protocol (SCTP) [RFC4960], and SCTP Partial Reliability Extension (SCTP-PR) [RFC3758], or Datagram Congestion Control Protocol (DCCP) [RFC4340] with its different congestion control types [RFC4341][RFC4342][RFC5622].

If used correctly, these more fully-featured transport protocols are not as "heavyweight" as often claimed. For example, the TCP algorithms have been continuously improved over decades, and have reached a level of efficiency and correctness that custom application-layer mechanisms will struggle to easily duplicate. In addition, many TCP implementations allow connections to be tuned by an application to its purposes. For example, TCP’s "Nagle" algorithm [RFC0896] can be disabled, improving communication latency at the expense of more frequent -- but still congestion-controlled -- packet transmissions. Another example is the TCP SYN cookie mechanism [RFC4987], which is available on many platforms. TCP with SYN cookies does not require a server to maintain per-connection state until the connection is established. TCP also requires the end that closes a connection to maintain the TIME-WAIT state that prevents delayed segments from one connection instance from interfering with a later one. Applications that are aware of and designed for this behavior can shift maintenance of the TIME-WAIT state to conserve resources by controlling which end closes a TCP connection [FABER]. Finally, TCP’s built-in capacity-probing and awareness of the maximum transmission unit supported by the path (PMTU) results in efficient data transmission that quickly compensates for the initial connection setup delay, in the case of transfers that exchange more than a few segments.

3.1. Congestion Control Guidelines

If an application or protocol chooses not to use a congestion-controlled transport protocol, it SHOULD control the rate at which it sends UDP datagrams to a destination host, in order to fulfill the requirements of [RFC2914]. It is important to stress that an application SHOULD perform congestion control over all UDP traffic it sends to a destination, independently from how it generates this traffic. For example, an application that forks multiple worker processes or otherwise uses multiple sockets to generate UDP
datagrams SHOULD perform congestion control over the aggregate traffic.

Several approaches to perform congestion control are discussed in the remainder of this section. The section describes generic topics with an intended emphasis on unicast and anycast [RFC1546] usage. Not all approaches discussed below are appropriate for all UDP-transmitting applications. Section 3.1.1 discusses congestion control options for applications that perform bulk transfers over UDP. Such applications can employ schemes that sample the path over several subsequent RTTs during which data is exchanged, in order to determine a sending rate that the path at its current load can support. Other applications only exchange a few UDP datagrams with a destination. Section 3.1.2 discusses congestion control options for such "low data-volume" applications. Because they typically do not transmit enough data to iteratively sample the path to determine a safe sending rate, they need to employ different kinds of congestion control mechanisms. Section 3.1.7 discusses congestion control considerations when UDP is used as a tunneling protocol. Section 4 provides additional recommendations for broadcast and multicast usage.

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

It is important to note that congestion control should not be viewed as an add-on to a finished application. Many of the mechanisms discussed in the guidelines below require application support to operate correctly. Application designers need to consider congestion control throughout the design of their application, similar to how they consider security aspects throughout the design process.

In the past, the IETF has also investigated integrated congestion control mechanisms that act on the traffic aggregate between two hosts, i.e., a framework such as the Congestion Manager [RFC3124], where active sessions may share current congestion information in a way that is independent of the transport protocol. Such mechanisms have currently failed to see deployment, but would otherwise simplify the design of congestion control mechanisms for UDP sessions, so that they fulfill the requirements in [RFC2914].
3.1.1. Bulk Transfer Applications

Applications that perform bulk transmission of data to a peer over UDP, i.e., applications that exchange more than a few UDP datagrams per RTT, SHOULD implement TCP-Friendly Rate Control (TFRC) [RFC5348], window-based TCP-like congestion control, or otherwise ensure that the application complies with the congestion control principles.

TFRC has been designed to provide both congestion control and fairness in a way that is compatible with the IETF’s other transport protocols. If an application implements TFRC, it need not follow the remaining guidelines in Section 3.1.1, because TFRC already addresses them, but SHOULD still follow the remaining guidelines in the subsequent subsections of Section 3.

Bulk transfer applications that choose not to implement TFRC or TCP-like windowing SHOULD implement a congestion control scheme that results in bandwidth use that competes fairly with TCP within an order of magnitude. Section 2 of [RFC3551] suggests that applications SHOULD monitor the packet loss rate to ensure that it is within acceptable parameters. Packet loss is considered acceptable if a TCP flow across the same network path under the same network conditions would achieve an average throughput, measured on a reasonable timescale, that is not less than that of the UDP flow. The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in timescale and throughput.

Finally, some bulk transfer applications may choose not to implement any congestion control mechanism and instead rely on transmitting across reserved path capacity. This might be an acceptable choice for a subset of restricted networking environments, but is by no means a safe practice for operation over the wider Internet. When the UDP traffic of such applications leaks out into unprovisioned Internet paths, it can significantly degrade the performance of other traffic sharing the path and even result in congestion collapse. Applications that support an uncontrolled or unadaptive transmission behavior SHOULD NOT do so by default and SHOULD instead require users to explicitly enable this mode of operation.

3.1.2. Low Data-Volume Applications

When applications that at any time exchange only a few UDP datagrams with a destination implement TFRC or one of the other congestion control schemes in Section 3.1.1, the network sees little benefit, because those mechanisms perform congestion control in a way that is only effective for longer transmissions.
Applications that at any time exchange only a few UDP datagrams with a destination SHOULD still control their transmission behavior by not sending on average more than one UDP datagram per round-trip time (RTT) to a destination. Similar to the recommendation in [RFC1536], an application SHOULD maintain an estimate of the RTT for any destination with which it communicates. Applications SHOULD implement the algorithm specified in [RFC6298] to compute a smoothed RTT (SRTT) estimate. They SHOULD also detect packet loss and exponentially back their retransmission timer off when a loss event occurs. When implementing this scheme, applications need to choose a sensible initial value for the RTT. This value SHOULD generally be as conservative as possible for the given application. TCP uses an initial value of 3 seconds [RFC6298], which is also RECOMMENDED as an initial value for UDP applications. SIP [RFC3261] and GIST [RFC5971] use an initial value of 500 ms, and initial timeouts that are shorter than this are likely problematic in many cases. It is also important to note that the initial timeout is not the maximum possible timeout -- the RECOMMENDED algorithm in [RFC6298] yields timeout values after a series of losses that are much longer than the initial value.

Some applications cannot maintain a reliable RTT estimate for a destination. The first case is that of applications that exchange too few UDP datagrams with a peer to establish a statistically accurate RTT estimate. Such applications MAY use a predetermined transmission interval that is exponentially backed-off when packets are lost. TCP uses an initial value of 3 seconds [RFC6298], which is also RECOMMENDED as an initial value for UDP applications. SIP [RFC3261] and GIST [RFC5971] use an interval of 500 ms, and shorter values are likely problematic in many cases. As in the previous case, note that the initial timeout is not the maximum possible timeout.

A second class of applications cannot maintain an RTT estimate for a destination, because the destination does not send return traffic. Such applications SHOULD NOT send more than one UDP datagram every 3 seconds, and SHOULD use an even less aggressive rate when possible. The 3-second interval was chosen based on TCP’s retransmission timeout when the RTT is unknown [RFC6298], and shorter values are likely problematic in many cases. Note that the sending rate in this case must be more conservative than in the two previous cases, because the lack of return traffic prevents the detection of packet loss, i.e., congestion, and the application therefore cannot perform exponential back-off to reduce load.

Applications that communicate bidirectionally SHOULD employ congestion control for both directions of the communication. For example, for a client-server, request-response-style application, clients SHOULD congestion-control their request transmission to a
server, and the server SHOULD congestion-control its responses to the clients. Congestion in the forward and reverse direction is uncorrelated, and an application SHOULD either independently detect and respond to congestion along both directions, or limit new and retransmitted requests based on acknowledged responses across the entire round-trip path.

3.1.3. Burst Mitigation and Pacing

UDP applications SHOULD provide mechanisms to regulate the bursts of transmission that the application may send to the network. Many TCP and SCTP implementations provide mechanisms that prevent a sender from generating long bursts at line-rate, since these are known to induce early loss to applications sharing a common network bottleneck. The use of pacing with TCP has also been shown to improve the coexistence of TCP flows with other flows.

Even low data-volume UDP flows may benefit from rate control, e.g., an application that sends three copies of a packet to improve robustness to loss is RECOMMENDED to pace out those three packets over several RTTs, to reduce the probability that all three packets will be lost due to the same congestion event.

3.1.4. Differentiated Services Model

An application using UDP can use the differentiated services 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. A DSCP may be chosen from a small set of fixed values (the class selector codepoints), 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 the 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.

[I-D.ietf-dart-dscp-rtp] 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.

3.1.5. QoS, Preprovisioned or Reserved Capacity

An application using UDP can use the integrated services QoS framework. These are usually available within controlled environments (e.g., within a single administrative domain or bilaterally agreed connection between domains). Applications intended for the Internet should not assume that QoS mechanisms are supported by the networks they use, and therefore need to provide congestion control, error recovery, etc. in case the actual network path does not provide provisioned service.

Some UDP applications are only expected to be deployed over network paths that use preprovisioned capacity or capacity reserved using dynamic provisioning, e.g., through the Resource Reservation Protocol (RSVP). Multicast applications are also used with preprovisioned capacity (e.g., IPTV deployments within access networks). These applications MAY choose not to implement any congestion control mechanism and instead rely on transmitting only on paths where the capacity is provisioned and reserved for this use. This might be an acceptable choice for a subset of restricted networking environments, but is by no means a safe practice for operation over the wider Internet.

If the traffic of such applications leaks out into unprovisioned Internet paths, it can significantly degrade the performance of other traffic sharing the path and even result in congestion collapse. For this reason, and to protect other applications sharing the same path, applications SHOULD deploy an appropriate circuit breaker, as described in Section 3.1.6. Applications that support an uncontrolled or unadaptive transmission behavior SHOULD NOT do so by default and SHOULD instead require users to explicitly enable this mode of operation.

Applications used in networks within a controlled environment may be able to exploit network management functions to detect whether they are causing congestion, and react accordingly.
3.1.6. 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. Later sections provide examples of cases where circuit breakers may or may not be desirable.


3.1.7. UDP Tunnels

One increasingly popular use of UDP is as a tunneling protocol, where a tunnel endpoint encapsulates the packets of another protocol inside UDP datagrams and transmits them to another tunnel endpoint, which decapsulates the UDP datagrams and forwards the original packets contained in the payload. Tunnels establish virtual links that appear to directly connect locations that are distant in the physical Internet topology and can be used to create virtual (private) networks. Using UDP as a tunneling protocol is attractive when the payload protocol is not supported by middleboxes that may exist along the path, because many middleboxes support transmission using UDP.

Well-implemented tunnels are generally invisible to the endpoints that happen to transmit over a path that includes tunneled links. On the other hand, to the routers along the path of a UDP tunnel, i.e., the routers between the two tunnel endpoints, the traffic that a UDP tunnel generates is a regular UDP flow, and the encapsulator and decapsulator appear as regular UDP-sending and -receiving applications. Because other flows can share the path with one or more UDP tunnels, congestion control needs to be considered.

Two factors determine whether a UDP tunnel needs to employ specific congestion control mechanisms -- first, whether the payload traffic is IP-based; second, whether the tunneling scheme generates UDP
traffic at a volume that corresponds to the volume of payload traffic carried within the tunnel.

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

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

The following guidelines define these possible cases in more detail:

1. A tunnel generates UDP traffic at a volume that corresponds to the volume of payload traffic, and the payload traffic is IP-based and congestion-controlled.

   This is arguably the most common case for Internet tunnels. In this case, the UDP tunnel SHOULD NOT employ its own congestion control mechanism, because congestion losses of tunneled traffic will already trigger an appropriate congestion response at the original senders of the tunneled traffic.

   Note that this guideline is built on the assumption that most IP-based communication is congestion-controlled. If a UDP tunnel is used for IP-based traffic that is known to not be congestion-controlled, the next set of guidelines applies.

2. A tunnel generates UDP traffic at a volume that corresponds to the volume of payload traffic, and the payload traffic is not known to be IP-based, or is known to be IP-based but not congestion-controlled.

   This can be the case, for example, when some link-layer protocols are encapsulated within UDP (but not all link-layer protocols; some are congestion-controlled). Because it is not known that congestion losses of tunneled non-IP traffic will trigger an appropriate congestion response at the senders, the UDP tunnel SHOULD employ an appropriate congestion control mechanism. Because tunnels are usually bulk-transfer applications as far as the intermediate routers are concerned, the guidelines in Section 3.1.1 apply.
3. A tunnel generates UDP traffic at a volume that does not correspond to the volume of payload traffic, independent of whether the payload traffic is IP-based or congestion-controlled. Examples of this class include UDP tunnels that send at a constant rate, increase their transmission rates under loss, for example, due to increasing redundancy when Forward Error Correction is used, or are otherwise unconstrained in their transmission behavior. These specialized uses of UDP for tunneling go beyond the scope of the general guidelines given in this document. The implementer of such specialized tunnels SHOULD carefully consider congestion control in the design of their tunneling mechanism and SHOULD consider use of a circuit breaker mechanism.

Designing a tunneling mechanism requires significantly more expertise than needed for many other UDP applications, because tunnels are usually intended to be transparent to the endpoints transmitting over them, so they need to correctly emulate the behavior of an IP link, e.g., handling fragmentation, generating and responding to ICMP messages, etc. At the same time, the tunneled traffic is application traffic like any other from the perspective of the networks the tunnel transmits over. This document only touches upon the congestion control considerations for implementing UDP tunnels; a discussion of other required tunneling behavior is out of scope.

3.2. Message Size Guidelines

IP fragmentation lowers the efficiency and reliability of Internet communication. The loss of a single fragment results in the loss of an entire fragmented packet, because even if all other fragments are received correctly, the original packet cannot be reassembled and delivered. This fundamental issue with fragmentation exists for both IPv4 and IPv6. In addition, some network address translators (NATs) and firewalls drop IP fragments. The network address translation performed by a NAT only operates on complete IP packets, and some firewall policies also require inspection of complete IP packets. Even with these being the case, some NATs and firewalls simply do not implement the necessary reassembly functionality, and instead choose to drop all fragments. Finally, [RFC4963] documents other issues specific to IPv4 fragmentation.

Due to these issues, an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the MTU of the path to the destination. Consequently, an application SHOULD either use the path MTU information provided by the IP layer or implement path MTU discovery itself [RFC1191][RFC1981][RFC4821] to determine whether the
Applications that do not follow this recommendation to do PMTU discovery SHOULD still avoid sending UDP datagrams that would result in IP packets that exceed the path MTU. Because the actual path MTU is unknown, such applications SHOULD fall back to sending messages that are shorter than the default effective MTU for sending (EMTU_S in [RFC1122]). For IPv4, EMTU_S is the smaller of 576 bytes and the first-hop MTU [RFC1122]. For IPv6, EMTU_S is 1280 bytes [RFC2460].

The effective PMTU for a directly connected destination (with no routers on the path) is the configured interface MTU, which could be less than the maximum link payload size. Transmission of minimum-sized UDP datagrams is inefficient over paths that support a larger PMTU, which is a second reason to implement PMTU discovery.

To determine an appropriate UDP payload size, applications MUST subtract the size of the IP header (which includes any IPv4 optional headers or IPv6 extension headers) as well as the length of the UDP header (8 bytes) from the PMTU size. This size, known as the MSS, can be obtained from the TCP/IP stack [RFC1122].

Applications that do not send messages that exceed the effective PMTU of IPv4 or IPv6 need not implement any of the above mechanisms. Note that the presence of tunnels can cause an additional reduction of the effective PMTU, so implementing PMTU discovery may be beneficial.

Applications that fragment an application-layer message into multiple UDP datagrams SHOULD perform this fragmentation so that each datagram can be received independently, and be independently retransmitted in the case where an application implements its own reliability mechanisms.

Packetization Layer Path MTU Discovery (PLPMTUD) [RFC4821] does not rely upon network support for ICMP messages and is therefore considered more robust than standard PMTUD. To operate, PLPMTUD requires changes to the way the transport is used, both to transmit probe packets, and to account for the loss or success of these probes. This updates not only the PMTU algorithm, it also impacts loss recovery, congestion control, etc. These updated mechanisms can be implemented within a connection-oriented transport (e.g., TCP, SCTP, DCCP), but are not a part of UDP. PLPMTUD therefore places additional design requirements on a UDP application that wishes to use this method.
3.3. Reliability Guidelines

Application designers are generally aware that UDP does not provide any reliability, e.g., it does not retransmit any lost packets. Often, this is a main reason to consider UDP as a transport. Applications that do require reliable message delivery MUST implement an appropriate mechanism themselves.

UDP also does not protect against datagram duplication, i.e., an application may receive multiple copies of the same UDP datagram, with some duplicates arriving potentially much later than the first. Application designers SHOULD verify that their application handles such datagram duplication gracefully, and may consequently need to implement mechanisms to detect duplicates. Even if UDP datagram reception triggers only idempotent operations, applications may want to suppress duplicate datagrams to reduce load.

Applications that require ordered delivery MUST reestablish datagram ordering themselves. The Internet can significantly delay some packets with respect to others, e.g., due to routing transients, intermittent connectivity, or mobility. This can cause reordering, where UDP datagrams arrive at the receiver in an order different from the transmission order.

It is important to note that the time by which packets are reordered or after which duplicates can still arrive can be very large. Even more importantly, there is no well-defined upper boundary here. [RFC0793] defines the maximum delay a TCP segment should experience -- the Maximum Segment Lifetime (MSL) -- as 2 minutes. No other RFC defines an MSL for other transport protocols or IP itself. The MSL value defined for TCP is conservative enough that it SHOULD be used by other protocols, including UDP. Therefore, applications SHOULD be robust to the reception of delayed or duplicate packets that are received within this 2-minute interval.

Instead of implementing these relatively complex reliability mechanisms by itself, an application that requires reliable and ordered message delivery SHOULD whenever possible choose an IETF standard transport protocol that provides these features.

3.4. Checksum Guidelines

The UDP header includes an optional, 16-bit one’s complement checksum that provides an integrity check. These checks are not strong from a coding or cryptographic perspective, and are not designed to detect physical-layer errors or malicious modification of the datagram [RFC3819]. Application developers SHOULD implement additional checks where data integrity is important, e.g., through a Cyclic Redundancy
Check (CRC) included with the data to verify the integrity of an entire object/file sent over the UDP service.

The UDP checksum provides a statistical guarantee that the payload was not corrupted in transit. It also allows the receiver to verify that it was the intended destination of the packet, because it covers the IP addresses, port numbers, and protocol number, and it verifies that the packet is not truncated or padded, because it covers the size field. It therefore protects an application against receiving corrupted payload data in place of, or in addition to, the data that was sent. More description of the set of checks performed using the checksum field are provided in Section 3.1 of [RFC6396].

Applications SHOULD enable UDP check sums. For IPv4, [RFC0768] permits an option to disable their use.

When UDP is used over IPv6, the UDP checksum is relied upon to protect both the IPv6 and UDP headers from corruption, and MUST be used as specified in [RFC2460], unless the requirements in [RFC6935] and [RFC6936] for use of UDP zero-checksum mode with a tunnel protocol are satisfied. The application MUST implement mechanisms and/or usage restrictions for this mode. These additional design requirements for using a zero IPv6 UDP checksum [RFC6936] are not present for IPv4, since the IPv4 header validates information that is not protected in an IPv6 packet. Key requirements apply to implementation and use of UDP zero-checksum mode for IPv6:

- Use of the UDP checksum with IPv6 MUST be the default configuration for all implementations [RFC6935]. The receiving endpoint MUST only allow the use of UDP zero-checksum mode for IPv6 on a UDP destination port that is specifically enabled.

- An application MUST comply with all implementation requirements specified in Section 4 of [RFC6936] and with usage requirements specified in Section 5 of [RFC6936].

- A UDP application MUST check that the source and destination IPv6 addresses are valid for any packets with a UDP zero-checksum and MUST discard any packet for which this check fails.

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.

3.4.1. UDP-Lite

A special class of applications can derive benefit from having partially-damaged payloads delivered, rather than discarded, when using paths that include error-prone links. Such applications can tolerate payload corruption and MAY choose to use the Lightweight User Datagram Protocol (UDP-Lite) [RFC3828] variant of UDP instead of basic UDP. Applications that choose to use UDP-Lite instead of UDP should still follow the congestion control and other guidelines described for use with UDP in Section 3.

UDP-Lite changes the semantics of the UDP "payload length" field to that of a "checksum coverage length" field. Otherwise, UDP-Lite is semantically identical to UDP. The interface of UDP-Lite differs from that of UDP by the addition of a single (socket) option that communicates a checksum coverage length value: at the sender, this specifies the intended checksum coverage, with the remaining unprotected part of the payload called the "error-insensitive part". By default, the UDP-Lite checksum coverage extends across the entire datagram. If required, an application may dynamically modify this length value, e.g., to offer greater protection to some messages. UDP-Lite always verifies that a packet was delivered to the intended destination, i.e., always verifies the header fields. Errors in the insensitive part will not cause a UDP datagram to be discarded by the destination. Applications using UDP-Lite therefore MUST NOT make assumptions regarding the correctness of the data received in the insensitive part of the UDP-Lite payload.

A UDP-Lite sender SHOULD select the minimum checksum coverage to include all sensitive payload information. For example, applications that use the Real-Time Protocol (RTP) [RFC3550] will likely want to protect the RTP header against corruption. Applications, where appropriate, MUST also introduce their own appropriate validity checks for protocol information carried in the insensitive part of the UDP-Lite payload (e.g., internal CRCs).

A UDP-Lite receiver MUST set a minimum coverage threshold for incoming packets that is not smaller than the smallest coverage used by the sender [RFC3828]. The receiver SHOULD select a threshold that is sufficiently large to block packets with an inappropriately short
coverage field. This may be a fixed value, or may be negotiated by an application. UDP-Lite does not provide mechanisms to negotiate the checksum coverage between the sender and receiver.

Applications can still experience packet loss when using UDP-Lite. The enhancements offered by UDP-Lite rely upon a link being able to intercept the UDP-Lite header to correctly identify the partial coverage required. When tunnels and/or encryption are used, this can result in UDP-Lite datagrams being treated the same as UDP datagrams, i.e., result in packet loss. Use of IP fragmentation can also prevent special treatment for UDP-Lite datagrams, and this is another reason why applications SHOULD avoid IP fragmentation (Section 3.2).

Current support for middlebox traversal using UDP-Lite is poor, because UDP-Lite uses a different IPv4 protocol number or IPv6 "next header" value than that used for UDP; therefore, few middleboxes are currently able to interpret UDP-Lite and take appropriate actions when forwarding the packet. This makes UDP-Lite less suited for applications needing general Internet support, until such time as UDP-Lite has achieved better support in middleboxes and endpoints.

3.5. Middlebox Traversal Guidelines

Network address translators (NATs) and firewalls are examples of intermediary devices ("middleboxes") that can exist along an end-to-end path. A middlebox typically performs a function that requires it to maintain per-flow state. For connection-oriented protocols, such as TCP, middleboxes snoop and parse the connection-management information and create and destroy per-flow state accordingly. For a connectionless protocol such as UDP, this approach is not possible. Consequently, middleboxes may create per-flow state when they see a packet that -- according to some local criteria -- indicates a new flow, and destroy the state after some period of time during which no packets belonging to the same flow have arrived.

Depending on the specific function that the middlebox performs, this behavior can introduce a time-dependency that restricts the kinds of UDP traffic exchanges that will be successful across the middlebox. For example, NATs and firewalls typically define the partial path on one side of them to be interior to the domain they serve, whereas the partial path on their other side is defined to be exterior to that domain. Per-flow state is typically created when the first packet crosses from the interior to the exterior, and while the state is present, NATs and firewalls will forward return traffic. Return traffic that arrives after the per-flow state has timed out is dropped, as is other traffic that arrives from the exterior.
Many applications that use UDP for communication operate across middleboxes without needing to employ additional mechanisms. One example is the Domain Name System (DNS), which has a strict request-response communication pattern that typically completes within seconds.

Other applications may experience communication failures when middleboxes destroy the per-flow state associated with an application session during periods when the application does not exchange any UDP traffic. Applications SHOULD be able to gracefully handle such communication failures and implement mechanisms to re-establish application-layer sessions and state.

For some applications, such as media transmissions, this re-synchronization is highly undesirable, because it can cause user-perceivable playback artifacts. Such specialized applications MAY send periodic keep-alive messages to attempt to refresh middlebox state. It is important to note that keep-alive messages are NOT RECOMMENDED for general use -- they are unnecessary for many applications and can consume significant amounts of system and network resources.

An application that needs to employ keep-alives to deliver useful service over UDP in the presence of middleboxes SHOULD NOT transmit them more frequently than once every 15 seconds and SHOULD use longer intervals when possible. No common timeout has been specified for per-flow UDP state for arbitrary middleboxes. NATs require a state timeout of 2 minutes or longer [RFC4787]. However, empirical evidence suggests that a significant fraction of currently deployed middleboxes unfortunately use shorter timeouts. The timeout of 15 seconds originates with the Interactive Connectivity Establishment (ICE) protocol [RFC5245]. When an application is deployed in a controlled network environment, the deployer SHOULD investigate whether the target environment allows applications to use longer intervals, or whether it offers mechanisms to explicitly control middlebox state timeout durations, for example, using Middlebox Communications (MIDCOM) [RFC3303], Next Steps in Signaling (NSIS) [RFC5973], or Universal Plug and Play (UPnP) [UPnP]. It is RECOMMENDED that applications apply slight random variations ("jitter") to the timing of keep-alive transmissions, to reduce the potential for persistent synchronization between keep-alive transmissions from different hosts.

Sending keep-alives is not a substitute for implementing a mechanism to recover from broken sessions. Like all UDP datagrams, keep-alives can be delayed or dropped, causing middlebox state to time out. In addition, the congestion control guidelines in Section 3.1 cover all UDP transmissions by an application, including the transmission of...
middlebox keep-alives. Congestion control may thus lead to delays or temporary suspension of keep-alive transmission.

Keep-alive messages are NOT RECOMMENDED for general use. They are unnecessary for many applications and may consume significant resources. For example, on battery-powered devices, if an application needs to maintain connectivity for long periods with little traffic, the frequency at which keep-alives are sent can become the determining factor that governs power consumption, depending on the underlying network technology. Because many middleboxes are designed to require keep-alives for TCP connections at a frequency that is much lower than that needed for UDP, this difference alone can often be sufficient to prefer TCP over UDP for these deployments. On the other hand, there is anecdotal evidence that suggests that direct communication through middleboxes, e.g., by using ICE [RFC5245], does succeed less often with TCP than with UDP. The trade-offs between different transport protocols -- especially when it comes to middlebox traversal -- deserve careful analysis.

UDP applications need to be designed understanding that there are many variants of middlebox behavior, and although UDP is connection-less, middleboxes often maintain state for each UDP flow. Using multiple flows can consume available state space and also can lead to changes in the way the middlebox handles subsequent packets (either to protect its internal resources, or to prevent perceived misuse). This has implications on applications that use multiple UDP flows in parallel, even on multiple ports Section 5.1.1.

4. Multicast UDP Usage Guidelines

This section complements Section 3 by providing additional guidelines that are applicable to multicast and broadcast usage of UDP.

Multicast and broadcast transmission [RFC1112] usually employ the UDP transport protocol, although they may be used with other transport protocols (e.g., UDP-Lite).

There are currently two models of multicast delivery: the Any-Source Multicast (ASM) model as defined in [RFC1112] and the Source-Specific Multicast (SSM) model as defined in [RFC4607]. ASM group members will receive all data sent to the group by any source, while SSM constrains the distribution tree to only one single source.

Specialized classes of applications also use UDP for IP multicast or broadcast [RFC0919]. The design of such specialized applications requires expertise that goes beyond simple, unicast-specific guidelines, since these senders may transmit to potentially very many receivers across potentially very heterogeneous paths at the same
time, which significantly complicates congestion control, flow control, and reliability mechanisms. This section provides guidance on multicast UDP usage.

Use of broadcast by an application is normally constrained by routers to the local subnetwork. However, use of tunneling techniques and proxies can and does result in some broadcast traffic traversing Internet paths. These guidelines therefore also apply to broadcast traffic.

The IETF has defined a reliable multicast framework [RFC3048] and several building blocks to aid the designers of multicast applications, such as [RFC3738] or [RFC4654]. Anycast senders must be aware that successive messages sent to the same anycast IP address may be delivered to different anycast nodes, i.e., arrive at different locations in the topology.

Most UDP tunnels that carry IP multicast traffic use a tunnel encapsulation with a unicast destination address. These MUST follow the same requirements as a tunnel carrying unicast data (see Section 3.1.7). There are deployment cases and solutions where the outer header of a UDP tunnel contains a multicast destination address, such as [RFC6513]. These cases are primarily deployed in controlled environments over reserved capacity, often operating within a single administrative domain, or between two domains over a bi-laterally agreed upon path with reserved bandwidth, and so congestion control is OPTIONAL, but circuit breaker techniques are still RECOMMENDED in order to restore some degree of service should the offered load exceed the reserved capacity (e.g., due to misconfiguration).

4.1. Multicast Congestion Control Guidelines

Unicast congestion-controlled transport mechanism are often not applicable to multicast distribution services, or simply do not scale to large multicast trees, since they require bi-directional communication and adapt the sending rate to accommodate the network conditions to a single receiver. In contrast, multicast distribution trees may fan out to massive numbers of receivers, which limits the scalability of an in-band return channel to control the sending rate, and the one-to-many nature of multicast distribution trees prevents adapting the rate to the requirements of an individual receiver. For this reason, generating TCP-compatible aggregate flow rates for Internet multicast data, either native or tunneled, is the responsibility of the application.

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

Multicast congestion control needs to consider the potential heterogeneity of both the multicast distribution tree and the receivers belonging to a group. Heterogeneity may manifest itself in some receivers experiencing more loss that others, higher delay, and/or less ability to respond to network conditions. Any multicast-enabled receiver may attempt to join and receive traffic from any group. This may imply the need for rate limits on individual receivers or the aggregate multicast service. Note there is no way at the transport layer to prevent a join message propagating to the next-hop router. A multicast congestion control method MAY therefore decide not to reduce the rate of the entire multicast group in response to a report received by a single receiver; instead it can decide to expel each congested receiver from the multicast group and to then distribute content to these congested receivers at a lower-rate using unicast congestion-control. Care needs to be taken when this action results in many flows being simultaneously transitioned, so that this does not result in excessive traffic exasperating congestion and potentially contributing to congestion collapse.

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

4.1.1. Bulk Transfer Multicast Applications

Applications that perform bulk transmission of data over a multicast distribution tree, i.e., applications that exchange more than a few UDP datagrams per RTT, SHOULD implement a method for congestion control. The currently RECOMMENDED IETF methods are: Asynchronous Layered Coding (ALC) [RFC5775], TCP-Friendly Multicast Congestion Control (TFMCC) [RFC4654], Wave and Equation Based Rate Control (WEBRC) [RFC3738], NACK-Oriented Reliable Multicast (NORM) transport protocol [RFC5740], File Delivery over Unidirectional Transport (FLUTE) [RFC6726], Real Time Protocol/Control Protocol (RTP/RTCP), [RFC3550].

An application can alternatively implement another congestion control schemes following the guidelines of [RFC2887] and utilizing the framework of [RFC3048]. Bulk transfer applications that choose not to implement, [RFC4654][RFC5775], [RFC3738], [RFC5740], [RFC6726], or [RFC3550] SHOULD implement a congestion control scheme that
results in bandwidth use that competes fairly with TCP within an order of magnitude.

Section 2 of [RFC3551] states that multimedia applications SHOULD monitor the packet loss rate to ensure that it is within acceptable parameters. Packet loss is considered acceptable if a TCP flow across the same network path under the same network conditions would achieve an average throughput, measured on a reasonable timescale, that is not less than that of the UDP flow. The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in timescale and throughput.

4.1.2. Low Data-Volume Multicast Applications

All the recommendations in Section 3.1.2 are also applicable to such multicast applications.

4.2. Message Size Guidelines for Multicast

A multicast application SHOULD NOT send UDP datagrams that result in IP packets that exceed the effective MTU as described in section 3 of [RFC6807]. Consequently, an application SHOULD either use the effective MTU information provided by the Population Count Extensions to Protocol Independent Multicast [RFC6807] or implement path MTU discovery itself (see Section 3.2) to determine whether the path to each destination will support its desired message size without fragmentation.

5. Programming Guidelines

The de facto standard application programming interface (API) for TCP/IP applications is the "sockets" interface [POSIX]. Some platforms also offer applications the ability to directly assemble and transmit IP packets through "raw sockets" or similar facilities. This is a second, more cumbersome method of using UDP. The guidelines in this document cover all such methods through which an application may use UDP. Because the sockets API is by far the most common method, the remainder of this section discusses it in more detail.

Although the sockets API was developed for UNIX in the early 1980s, a wide variety of non-UNIX operating systems also implement it. The sockets API supports both IPv4 and IPv6 [RFC3493]. The UDP sockets API differs from that for TCP in several key ways. Because application programmers are typically more familiar with the TCP sockets API, this section discusses these differences. [STEVENS] provides usage examples of the UDP sockets API.
UDP datagrams may be directly sent and received, without any connection setup. Using the sockets API, applications can receive packets from more than one IP source address on a single UDP socket. Some servers use this to exchange data with more than one remote host through a single UDP socket at the same time. Many applications need to ensure that they receive packets from a particular source address; these applications MUST implement corresponding checks at the application layer or explicitly request that the operating system filter the received packets.

If a client/server application executes on a host with more than one IP interface, the application SHOULD send any UDP responses with an IP source address that matches the IP destination address of the UDP datagram that carried the request (see [RFC1122], Section 4.1.3.5). Many middleboxes expect this transmission behavior and drop replies that are sent from a different IP address, as explained in Section 3.5.

A UDP receiver can receive a valid UDP datagram with a zero-length payload. Note that this is different from a return value of zero from a read() socket call, which for TCP indicates the end of the connection.

Many operating systems also allow a UDP socket to be connected, i.e., to bind a UDP socket to a specific pair of addresses and ports. This is similar to the corresponding TCP sockets API functionality. However, for UDP, this is only a local operation that serves to simplify the local send/receive functions and to filter the traffic for the specified addresses and ports. Binding a UDP socket does not establish a connection -- UDP does not notify the remote end when a local UDP socket is bound. Binding a socket also allows configuring options that affect the UDP or IP layers, for example, use of the UDP checksum or the IP Timestamp option. On some stacks, a bound socket also allows an application to be notified when ICMP error messages are received for its transmissions [RFC1122].

UDP provides no flow-control, i.e., the sender at any given time does not know whether the receiver is able to handle incoming transmissions. This is another reason why UDP-based applications need to be robust in the presence of packet loss. This loss can also occur within the sending host, when an application sends data faster than the line rate of the outbound network interface. It can also occur on the destination, where receive calls fail to return all the data that was sent when the application issues them too infrequently (i.e., such that the receive buffer overflows). Robust flow control mechanisms are difficult to implement, which is why applications that need this functionality SHOULD consider using a full-featured transport protocol such as TCP.
When an application closes a TCP, SCTP or DCCP socket, the transport protocol on the receiving host is required to maintain TIME-WAIT state. This prevents delayed packets from the closed connection instance from being mistakenly associated with a later connection instance that happens to reuse the same IP address and port pairs. The UDP protocol does not implement such a mechanism. Therefore, UDP-based applications need to be robust in this case. One application may close a socket or terminate, followed in time by another application receiving on the same port. This later application may then receive packets intended for the first application that were delayed in the network.

5.1. Using UDP Ports

The rules procedures for the management of the Service Name and Transport Protocol Port Number Registry are specified in [RFC6335]. Recommendations for use of UDP ports are provided in [I-D.ietf-tsvwg-port-use].

A UDP sender SHOULD NOT use a zero source port value. 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. TCP commonly uses source port randomization for this reason [RFC6056]. Setting a "randomized" source port also helps provide greater assurance that reported ICMP errors originate from network systems on the path used by a particular flow.

A UDP receiver SHOULD NOT bind to port zero. Applications SHOULD implement corresponding receiver checks at the application layer or explicitly request that the operating system filter the received packets to prevent receiving packets with an arbitrary port. This measure is designed to provide additional protection from data injection attacks from an off-path source (where the port values may not be known). Although the source port value is often not directly used in multicast applications, this should still be set to a random or predetermined value.

The UDP port number fields have been used as a basis to design load-balancing solutions for IPv4. This approach has also been leveraged for IPv6 [RFC6438], but for IPv6 the "flow label" [RFC6437] may also be used as a basis for entropy for load balancing. This use of the flow label for load balancing is consistent with the intended use, although further clarity was needed to ensure the field can be consistently used for this purpose. Therefore, an updated IPv6 flow label [RFC6437] and ECMP routing [RFC6438] usage were specified. Router vendors are encouraged to start using the flow label as a part of the flow hash, providing support for IP-level ECMP without requiring use of UDP. The end-to-end use of flow labels for load
balancing is a long-term solution. Even if the usage of the flow label has been clarified, there will be a transition time before a significant proportion of endpoints start to assign a good quality flow label to the flows that they originate. The use of load balancing using the transport header fields will likely continue until widespread deployment is finally achieved.

5.1.1. Applications using Multiple UDP Ports

A single application may exchange several types of data. In some cases, this may require multiple UDP flows (e.g., multiple sets of flows, identified by different five-tuples). [RFC6335] recommends applications developers not to apply to IANA to be assigned multiple well-known ports (user or system). This does not discuss the implications of using multiple flows with the same well-known port or pairs of dynamic ports (e.g., identified by a service name or signaling protocol).

Use of multiple flows can impact the network in several ways:

- 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 can not be assumed both follow the same path (e.g., when ECMP is used, traffic is intentionally hashed onto different parallel paths based on the port numbers).

- Using several flows can also increase the occupancy of a binding or lookup table in a middlebox (e.g., NAPT or Firewall) which may cause the device to change the way it manages the flow state.

- Further, using excessive numbers of flows can degrade the ability of congestion control to react to congestion events, unless the congestion state is shared between all flows in a session.

Therefore, applications MUST NOT assume consistent behavior of middleboxes when multiple UDP flows are used; many devices respond differently as the number of ports used increases. Using multiple flows with different QoS requirements requires applications to verify that the expected performance is achieved using each individual flow (five-tuple), see Section 3.1.5.
5.2. ICMP Guidelines

Applications can utilize information about ICMP error messages that the UDP layer passes up for a variety of purposes [RFC1122]. Applications SHOULD appropriately validate the payload of ICMP messages to ensure these are received in response to transmitted traffic (i.e., a reported error condition that corresponds to a UDP datagram actually sent by the application). This requires context, such as local state about communication instances to each destination, that although readily available in connection-oriented transport protocols is not always maintained by UDP-based applications. Note that not all platforms have the necessary APIs to support this validation, and some platforms already perform this validation internally before passing ICMP information to the application.

Any application response to ICMP error messages SHOULD be robust to temporary routing failures, e.g., transient ICMP "unreachable" messages should not normally cause a communication abort.

6. Security Considerations

UDP does not provide communications security. Applications that need to protect their communications against eavesdropping, tampering, or message forgery SHOULD employ end-to-end security services provided by other IETF protocols. Applications that respond to short requests with potentially large responses are vulnerable to amplification attacks, and SHOULD authenticate the sender before responding. The source IP address of a request is not a useful authenticator, because it can easily be spoofed.

One option of securing UDP communications is with IPsec [RFC4301], which can provide authentication for flows of IP packets through the Authentication Header (AH) [RFC4302] and encryption and/or authentication through the Encapsulating Security Payload (ESP) [RFC4303]. Applications use the Internet Key Exchange (IKE) [RFC5246] to configure IPsec for their sessions. Depending on how IPsec is configured for a flow, it can authenticate or encrypt the UDP headers as well as UDP payloads. If an application only requires authentication, ESP with no encryption but with authentication is often a better option than AH, because ESP can operate across middleboxes. An application that uses IPsec requires the support of an operating system that implements the IPsec protocol suite.

Although it is possible to use IPsec to secure UDP communications, not all operating systems support IPsec or allow applications to easily configure it for their flows. A second option of securing UDP communications is through Datagram Transport Layer Security (DTLS)
DTLS provides communication privacy by encrypting UDP payloads. It does not protect the UDP headers. Applications can implement DTLS without relying on support from the operating system.

Many other options for authenticating or encrypting UDP payloads exist. For example, the GSS-API security framework [RFC2743] or Cryptographic Message Syntax (CMS) [RFC5652] could be used to protect UDP payloads. The IETF standard for securing RTP [RFC3550] communication sessions over UDP is the Secure Real-time Transport Protocol (SRTP) [RFC3711]. In some applications, a better solution is to protect larger stand-alone objects, such as files or messages, instead of individual UDP payloads. In these situations, CMS [RFC5652], S/MIME [RFC5751] or OpenPGP [RFC4880] could be used. In addition, there are many non-IETF protocols in this area.

Like congestion control mechanisms, security mechanisms are difficult to design and implement correctly. It is hence RECOMMENDED that applications employ well-known standard security mechanisms such as DTLS or IPsec, rather than inventing their own.

The Generalized TTL Security Mechanism (GTSM) [RFC5082] may be used with UDP applications (especially when the intended endpoint is on the same link as the sender). This is a lightweight mechanism that allows a receiver to filter unwanted packets.

In terms of congestion control, [RFC2309] and [RFC2914] discuss the dangers of congestion-unresponsive flows to the Internet. [I-D.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.

7. Summary

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

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUST tolerate a wide range of Internet path conditions</td>
<td>3</td>
</tr>
<tr>
<td>SHOULD use a full-featured transport (TCP, SCTP, DCCP)</td>
<td></td>
</tr>
</tbody>
</table>

<p>| SHOULD control rate of transmission | 3.1 |
| SHOULD perform congestion control over all traffic | |
| for bulk transfers, | 3.1.1 |
| SHOULD consider implementing TFRC | |
| else, SHOULD in other ways use bandwidth similar to TCP | |
| for non-bulk transfers, | 3.1.2 |
| SHOULD measure RTT and transmit max. 1 datagram/RTT | |
| else, SHOULD send at most 1 datagram every 3 seconds | |
| SHOULD back-off retransmission timers following loss | |
| for tunnels carrying IP Traffic, | 3.1.7 |
| SHOULD NOT perform congestion control | |
| for non-IP tunnels or rate not determined by traffic, | 3.1.7 |
| SHOULD perform congestion control | |
| SHOULD NOT send datagrams that exceed the PMTU, i.e., | 3.2 |
| SHOULD discover PMTU or send datagrams &lt; 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 | |</p>
<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHOULD enable IPv4 UDP checksum</td>
<td>3.4</td>
</tr>
<tr>
<td>SHOULD enable IPv6 UDP checksum; Specific application mechanisms are REQUIRED if a zero IPv6 UDP checksum is used.</td>
<td></td>
</tr>
<tr>
<td>else, MAY use UDP-Lite with suitable checksum coverage</td>
<td>3.4.1</td>
</tr>
<tr>
<td>SHOULD NOT always send middlebox keep-alives</td>
<td>3.5</td>
</tr>
<tr>
<td>MAY use keep-alives when needed (min. interval 15 sec)</td>
<td></td>
</tr>
<tr>
<td>MUST check IP source address</td>
<td>5</td>
</tr>
<tr>
<td>and, for client/server applications</td>
<td></td>
</tr>
<tr>
<td>SHOULD send responses from src address matching request</td>
<td></td>
</tr>
<tr>
<td>SHOULD use standard IETF security protocols when needed</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 1: Summary of recommendations

8. IANA Considerations

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

This document raises no IANA considerations.

9. Acknowledgments

The middlebox traversal guidelines in Section 3.5 incorporate ideas from Section 5 of [I-D.ford-behave-app] by Bryan Ford, Pyda Srisuresh, and Dan Kegel.
10. References

10.1. Normative References


10.2. Informative References


[RFC6807] Farinacci, D., Shepherd, G., Venaas, S., and Y. Cai,
"Population Count Extensions to Protocol Independent
Multicast (PIM)", RFC 6807, December 2012.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and
UDP Checksums for Tunneled Packets", RFC 6935, April 2013.

for the Use of IPv6 UDP Datagrams with Zero Checksums",
RFC 6936, April 2013.

Kivinen, "Internet Key Exchange Protocol Version 2
(IKEv2)", STD 79, RFC 7296, October 2014.

[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
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 [I-D.ietf-mpls-in-udp], 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) [[I-D.ietf-mpls-in-udp].

MPLS-in-UDP endpoints must 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 as 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-01:

- 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
URL: 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
RFC 2872 defines an Resource Reservation Protocol (RSVP) object for application identifiers. This document uses that App-ID and gives implementers specific guidelines for differing voice and video stream identifications to nodes along a reservation path, creating specific profiles for voice and video session identification.

Status of this Memo

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

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

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

This Internet-Draft will expire on July 4, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
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 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 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PE Length (8)                      | P-type = AUTH_APP          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Attribute Length                    | A-type = POLICY_LOCATOR    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sub-type = ASCII_DN                 | Application name as ASCII string |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                   (e.g. SAP.EXE)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

In line with how this policy element is constructed in RFC 2872, the A-type will remain "POLICY_LOCATOR".

The P-type field is first created in [RFC2752]. This document uses the existing P-type "AUTH_APP" for application traffic class.

The first Sub-type will be mandatory for every profile within this document, and will be "ASCII_DN". No other Sub-types are defined by any profile within this document, but MAY be included by individual implementations - and MUST be ignored if not understood by receiving implementations along the reservation path.
RFC 2872 states the #1 sub-element from RFC 2872 as the "identifier that uniquely identifies the application vendor", which is optional to include. This document modifies this vendor limitation so that the identifier need only be unique - and not limited to an application vendor (identifier). For example, this specification now allows an RFC that defines an industry recognizable term or string to be a valid identifier. For example, a term or string taken from another IETF document, such as "conversational" or "avconf" from [ID-TRAF-CLASS]. This sub-element is still optional to include.

The following subsections will define the values within the above template into specific profiles for voice and video identification.

3. The Voice and Video Application-ID Profiles

This section contains the elements of the Application ID policy object which is used to signal the application classes defined in [ID-TRAF-CLASS].

3.1 The Broadcast Profiles

Broadcast profiles are for minimally buffered one-way streaming flows, such as video surveillance, or Internet based concerts or non-VOD TV broadcasts such as live sporting events.

This document creates Broadcast profiles for

- Broadcast IPTV for audio and video
- Broadcast Live-events for audio and video
- Broadcast Surveillance for audio and video

Here is an example profile for identifying Broadcast Video-Surveillance

AUTH_APP, POLICY_LOCATOR, ASCII_DN,
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.video.surveillance, VER="

[Editor’s Note: "rfcXXXX" will be replaced with the RFC number assigned to the [ID-TRAF-CLASS] reference. This ‘note’ should be removed during the RFC-Editor review process.]

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value at this time.
3.2 The Realtime Interactive Profiles

Realtime Interactive profiles are for on-line gaming, and both remote and virtual avconf applications, in which the timing is particularly important towards the feedback to uses of these applications. This traffic type will generally not be UDP based, with minimal tolerance to RTT delays.

This document creates Realtime Interactive profiles for
- Realtime-Interactive Gaming
- Realtime-Interactive Remote-Desktop
- Realtime-Interactive Virtualized-Desktop

Here is the profile for identifying Realtime-Interactive Gaming

AUTH_APP, POLICY_LOCATOR, ASCII_DN,
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=realtime-interactive.gaming, VER="

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.3 The Multimedia Conferencing Profiles

There will be Multimedia Conferencing profiles for presentation data, application sharing and whiteboarding, where these applications will most often be associated with a larger Conversational (audio and/or audio/video) conference. Timing is important, but some minimal delays are acceptable, unlike the case for Realtime-Interactive traffic.

This document creates Multimedia-Conferencing profiles for
- Multimedia-Conferencing presentation-data
- Multimedia-Conferencing presentation-video
- Multimedia-Conferencing presentation-audio
- Multimedia-Conferencing application-sharing
- Multimedia-Conferencing whiteboarding

Here is the profile for identifying Multimedia-Conferencing Application-sharing

AUTH_APP, POLICY_LOCATOR, ASCII_DN,
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=multimedia-conferencing.application-sharing, VER="

Where the Globally Unique Identifier (GUID) indicates the RFC reference that created this well-known string [ID-TRAF-CLASS], the
APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.4 The Multimedia Streaming Profiles

Multimedia Streaming profiles are for more significantly buffered one-way streaming flows than Broadcast profiles. These include...

This document creates Multimedia Streaming profiles for

- Multimedia-Streaming multiplex
- Multimedia-Streaming webcast

Here is the profile for identifying Multimedia Streaming webcast

AUTH_APP, POLICY_LOCATOR, ASCII_DN,
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=multimedia-streaming.webcast, VER="

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.5 The Conversational Profiles

Conversational category is for realtime bidirectional communications, such as voice or video, and is the most numerous due to the choices of application with or without adjectives. The number of profiles is then doubled because there needs to be one for unadmitted and one for admitted. The IANA section lists all that are currently proposed for registration at this time, therefore there will not be an exhaustive list provided in this section.

This document creates Conversational profiles for

- Conversational Audio
- Conversational Audio Admitted
- Conversational Video
- Conversational Video Admitted
- Conversational Audio Avconf
- Conversational Audio Avconf Admitted
- Conversational Video Avconf
- Conversational Video Avconf Admitted
- Conversational Audio Immersive
- Conversational Audio Immersive Admitted
- Conversational Video Immersive
- Conversational Video Immersive Admitted

Here is an example profile for identifying Conversational Audio:
AUTH_APP, POLICY_LOCATOR, ASCII_DN,
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.audio, VER="

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAFF-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-TRAFF-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="
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,
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


7.2. Informative References


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.
DSCP and other packet markings for RTCWeb QoS

draft-ietf-tsvwg-rtcweb-qos-03

Abstract

Many networks, such as service provider and enterprise networks, can provide treatment for individual packets based on Differentiated Services Code Points (DSCP) on a per-hop basis. This document provides the recommended DSCP values for browsers to use for various classes of traffic.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 16, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
1. Introduction

Differentiated Services Code Points (DSCP) [RFC2474] style packet marking can help provide QoS in some environments. There are many use cases where such marking does not help, but it seldom makes things worse if packets are marked appropriately. In other words, if too many packets, say all audio or all audio and video, are marked for a given network condition then it can prevent desirable results. Either too much other traffic will be starved, or there is not enough capacity for the preferentially marked packets (i.e., audio and/or video).

This specification proposes how WebRTC applications can mark packets. This specification does not contradict or redefine any advice from previous IETF RFCs but simply provides a simple set of recommendations for implementers based on the previous RFCs.

There are some environments where priority markings frequently help. These include:


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.

Traditionally DSCP values have been thought of as being site specific, with each site selecting its own code points for each QoS level. However in the RTCWeb use cases, the browsers need to set them to something when there is no site specific information. In this document, "browsers" is used synonymously with "interactive User Agent" as defined in the HTML specification, [W3C.WD-html-20110525]. This document describes a reasonable default set of DSCP code point values drawn from existing RFCs and common usage. These code points are solely defaults.

This specification defines some inputs that the browser in an WebRTC application can look at to determine how to set the various packet markings and defines the mapping from abstract QoS policies (data type, priority level) to those packet markings.

2. Relation to Other Standards

This document exists as a complement to [I-D.ietf-dart-dscp-rtp], which describes the interaction between DSCP and real-time communications. It 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.

This specification does not change or override the advice in any other standards about setting packet markings. It simply provides a summary of them and provides the context of how they relate in the RTCWeb context. In some cases, such as DSCP where the normative RFC leaves open multiple options from which to choose, this clarifies which choice should be used in the RTCWeb context. This document also specifies the inputs that are needed by the browser to provide to the media engine.

The DSCP value set by the endpoint is not always trusted by the network. Therefore, 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) which provides basic best effort service. The mitigation for such action is through an authorization mechanism. Such authorization mechanism is outside the scope of this document.

3. Terminology

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as described in [RFC2119].
4. Inputs

The below uses the concept of a media flow, however this is usually not equivalent to a transport flow, i.e. as defined by a 5-tuple (source address, destination address, source port, destination port, and protocol). Instead each media flow contains all the packets associated with an independent media entity within one 5-tuple. There may be multiple media flows within the same 5-tuple. These media flows might consist of different media types and have different priorities. The following are the inputs that the browser provides to the media engine:

- **Data Type**: The browser provides this input as it knows if the flow is audio, interactive video with or without audio, non-interactive video with or without audio, or data.
- **Priority**: Another input is the relative treatment of the flow within that data type. Many applications have multiple media flows of the same data type and often some are more important than others. Likewise, in a video conference where the flows in the conference is of the same data type but contains different media types, the flow for audio may be more important than the video flow. JavaScript applications can tell the browser whether a particular media flow is high, medium, low or very low importance to the application.

[I-D.ietf-rtcweb-transports] defines in more detail what an individual media flow is within the WebRTC context.

5. DSCP Mappings

Below is a table of DSCP markings for each data type of interest to RTCWeb. These DSCP values for each data type listed are a reasonable default set of code point values taken from [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. The mitigation for such action is through an authorization mechanism. Such authorization mechanism is outside the scope of this document.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td>CS1 (8)</td>
<td>DF</td>
<td>EF (46)</td>
<td>EF (46)</td>
</tr>
<tr>
<td>Interactive Video with or Audio</td>
<td>CS1</td>
<td>DF</td>
<td>AF42, AF43</td>
<td>AF41,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>---</td>
<td>---</td>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td>without audio</td>
<td>(8)</td>
<td>(0)</td>
<td>(36, 38)</td>
<td>AF42 (34, 36)</td>
</tr>
<tr>
<td>Non-Interactive Video with or without audio</td>
<td>CS1</td>
<td>DF</td>
<td>AF32, AF33</td>
<td>AF31, AF32 (26, 28)</td>
</tr>
<tr>
<td></td>
<td>(8)</td>
<td>(0)</td>
<td>(28, 30)</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>CS1</td>
<td>DF</td>
<td>AF1x (10, 12, 14)</td>
<td>AF2x (18, 20, 22)</td>
</tr>
<tr>
<td></td>
<td>(8)</td>
<td>(0)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1

The columns "very low", "low", "Medium" and "high" are the priority levels. This priority value signifies the relative priority of the media flow within the application and is provided to the browser to assist it in selecting the DSCP value. The browser SHOULD first select the data type of the media flow. Within the data type, the priority of the media flow SHOULD be selected. All packets within a media flow SHOULD have the same priority. In some cases, the selected cell may have multiple DSCP values, such as AF41 and AF42. These offer different drop precedences. One may select difference drop precedences for the different packets in the media flow. Therefore, all packets in the stream SHOULD be marked with the same priority but can have difference drop precedences.

The combination of data type and priority provides specificity and helps in selecting the right DSCP value for the media flow. In some cases, the different drop precedence values provides additional granularity in classifying packets within a media flow. For example, in a video conference, the video media flow may be medium priority. If so, either AF42 or AF43 may be selected. If the I frames in the stream are more important than the P frames then the I frames can be marked with AF42 and the P frames marked with AF43.

The above table assumes that packets marked with CS1 is treated as "less than best effort". However, the treatment of CS1 is implementation dependent. If an implementation treats CS1 as other than "less than best effort", then the priority of the packets may be changed from what is intended.
If a packet enters a QoS domain that has no support for the above defined Data Types/Application (service) classes, then the network node at the edge will remark the DSCP value based on policies. Subsequently, if the packet enters a QoS domain that supports a larger number of Data types/Application (service) classes, 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.

6. Security Considerations

This specification does not add any additional security implication other than the normal application use of DSCP. For security implications on use of DSCP, please refer to Section 6 of RFC 4594. 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]. However, the parts of that RFC used by this specification are sufficiently stable for this downward reference.

9. Acknowledgements

Thanks To David Black, Magnus Westerland, Paolo Severini, Jim Hasselbrook, Joe Marcus, and Erik Nordmark for their help.

10. 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. This document is now being submitted as individual submission to the TSVWG with the hope that WG will select it as a WG draft and move it forward to an RFC.
11. References

11.1. Normative References

[I-D.ietf-dart-dscp-rtp]
Black, D. and P. Jones, "Differentiated Services (DiffServ) and Real-time Communication", draft-ietf-dart-dscp-rtp-10 (work in progress), November 2014.

[I-D.ietf-rtcweb-security]


11.2. Informative References


Authors’ Addresses

Subha Dhesikan
Cisco Systems

Email: sdhesika@cisco.com
Abstract

The Stream Control Transmission Protocol (SCTP) is a transport protocol originally defined to run on top of the network protocols IPv4 or IPv6. This document specifies how SCTP can be used on top of the Datagram Transport Layer Security (DTLS) protocol. Using the encapsulation method described in this document, SCTP is unaware of the protocols being used below DTLS; hence explicit IP addresses cannot be used in the SCTP control chunks. As a consequence, the SCTP associations carried over DTLS can only be single homed.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 28, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
# 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.
]

## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Overview</td>
<td>2</td>
</tr>
<tr>
<td>2. Conventions</td>
<td>3</td>
</tr>
<tr>
<td>3. Encapsulation and Decapsulation Procedure</td>
<td>3</td>
</tr>
<tr>
<td>4. General Considerations</td>
<td>3</td>
</tr>
<tr>
<td>5. DTLS Considerations</td>
<td>4</td>
</tr>
<tr>
<td>6. SCTP Considerations</td>
<td>5</td>
</tr>
<tr>
<td>7. IANA Considerations</td>
<td>6</td>
</tr>
<tr>
<td>8. Security Considerations</td>
<td>6</td>
</tr>
<tr>
<td>9. Acknowledgments</td>
<td>7</td>
</tr>
<tr>
<td>10. References</td>
<td>7</td>
</tr>
<tr>
<td>Appendix A. NOTE to the RFC-Editor</td>
<td>9</td>
</tr>
<tr>
<td>Authors’ Addresses</td>
<td>9</td>
</tr>
</tbody>
</table>
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:

- A DTLS connection MUST be established before an SCTP association can be set up.
- 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.
- 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.
- 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.
- 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.
- 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-sctpndata] 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


10.2. Informative References


[I-D.ietf-rtcweb-data-channel]

[I-D.ietf-tsvwg-sctp-prpolicies]

[I-D.ietf-tsvwg-sctp-ndata]

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_iertf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas  02420
FI

Email: Salvatore.Loreto@ericsson.com
SCTP-PF: Quick Failover Algorithm in SCTP
draft-ietf-tsvwg-sctp-failover-10.txt

Abstract

One of the major advantages of SCTP is the support of multi-homed communication. A multi-homed SCTP end-point has the ability to withstand network failures by migrating the traffic from an inactive network to an active one. However, if the failover operation as specified in RFC4960 is followed, there can be a significant delay in the migration to the active destination addresses, thus severely reducing the effectiveness of the SCTP failover operation.

This document complements RFC4960 by the introduction of a new path state, the Potentially Failed (PF) path state, and an associated new failover operation to apply during a network failure. The algorithm defined is called SCTP Potentially Failed Algorithm, SCTP-PF for short. In addition, the document complements RFC4960 by introducing alternative switchover operation modes for the data transfer path management after the recovery of a failed primary path. These modes can allow improvements in the performance of the operation in some network environments. The implementation of the additional switchover operation modes is an optional part of SCTP-PF.

The procedures defined in the document require only minimal modifications to the current specification. The procedures are sender-side only and do not impact the SCTP receiver.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the
document authors. All rights reserved.

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

Table of Contents

1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
2. Conventions and Terminology . . . . . . . . . . . . . . . . . 4
3. Issues with the SCTP Path Management . . . . . . . . . . . . 4
4. SCTP with Potentially-Failed Destination State (SCTP-PF) . . 5
  4.1. SCTP-PF Concept . . . . . . . . . . . . . . . . . . . . . . . 5
  4.2. Specification of the SCTP-PF Algorithm . . . . . . . . . . 6
    4.2.1. Dormant State Operation . . . . . . . . . . . . . . . . 10
  4.3. Permanent Failover . . . . . . . . . . . . . . . . . . . . . 12
    4.3.1. Background . . . . . . . . . . . . . . . . . . . . . . . 12
    4.3.2. Permanent Failover Algorithm . . . . . . . . . . . . 12
5. Socket API Considerations . . . . . . . . . . . . . . . . . . . 13
  5.1. Support for the Potentially Failed Path State . . . . . . 14
  5.2. Peer Address Thresholds (SCTP_PEER_ADDR_THLDS) Socket
      Option . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
  5.3. Exposing the Potentially Failed Path State
      (SCTP_EXPOSE_POTENTIALLY_FAILED_STATE) Socket Option . . 16
6. Security Considerations . . . . . . . . . . . . . . . . . . . . . 16
7. IANA Considerations . . . . . . . . . . . . . . . . . . . . . . 17
8. Proposed Change of Status (to be Deleted before Publication) 17
9. References . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
  9.1. Normative References . . . . . . . . . . . . . . . . . . . . 17

1. Introduction

The Stream Control Transmission Protocol (SCTP) as specified in [RFC4960] supports multihoming at the transport layer -- an SCTP endpoint can bind to multiple IP addresses. SCTP’s multihoming 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. The sender fails over to an alternate active destination address only after failure detection. Until detecting the failover, the sender continues to transmit data on the failed path, which degrades the SCTP performance. Concurrent Multipath Transfer (CMT) [IYENGAR06] is an proposed extension to SCTP that allows the sender to transmit data on multiple paths simultaneously. Research [NATARAJAN09] shows that the current failure detection procedure worsens CMT performance during failover and can be significantly improved by employing a better failover algorithm.

This document specifies an alternative failure detection and failover procedure, the SCTP Potentially Failed algorithm, that improves the performance of SCTP multi-homed operation during a failover.

For multi-homed SCTP the operation after the recovery of a failed path equally well impacts the performance of the protocol. With the procedures specified in [RFC4960], SCTP will, after a failover from the primary path, switch back to the primary path for data transfer as soon as this path becomes available again. From a performance perspective, as confirmed in research [CARO02], such a switchback of the data transmission path is not optimal in general. As an optional alternative to the switchback operation of [RFC4960], this document specifies the Permanent Failover procedures proposed by [CARO02].

Additional discussion for alternative approaches that do not require modifications to [RFC4960], as well as discussion of path bouncing effects that might be caused by frequent switchover, are provided in the Appendices.

While the Potentially Failed algorithm primarily is motivated for improvement of the SCTP 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.

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. Issues with the SCTP Path Management

This section describes issues in the SCTP as specified in [RFC4960] to be fixed by the approach described in this document.

An SCTP endpoint can support multiple IP addresses. Each SCTP endpoint exchanges the list of its usable addresses during the initial negotiation with its peer. Then the endpoints select one address from the peer’s list and use this as the primary destination address. During normal transmission, an SCTP endpoint sends all user data to the primary destination address. Also, it sends packets containing a HEARTBEAT chunk to all idle destination addresses at a certain interval to check the reachability of these destination addresses. Idle destination addresses normally include all non-primary destination addresses.

If a sender has multiple active destination addresses, it can retransmit data to an non-primary destination address, if the transmission to the primary times out.

When a sender receives an acknowledgment for DATA or HEARTBEAT chunks sent to one of the destination addresses, it considers that destination address to be active and clears the error counter for the destination address. If it fails to receive acknowledgments, the error count for the destination address is increased. If the error counter exceeds the tunable protocol parameter Path.Max.Retrans (PMR), the SCTP endpoint considers the destination address to be inactive.

The failover process of SCTP is initiated when the primary path becomes inactive (the error counter for the primary path exceeds Path.Max.Retrans). If the primary path is marked inactive, SCTP chooses a new destination address from one of the active destinations and starts using this as the destination address for sending data. If the primary path becomes active again, SCTP reverts to using the
primary destination address for subsequent data transmissions and stop using the non-primary one.

One issue with this failover process defined in [RFC4960] is that it usually takes a significant amount of time before SCTP switches to the new destination address. Let’s say the primary path on a multi-homed host becomes unavailable and the RTO value for the primary path at that time is around 1 second, it usually takes over 60 seconds before SCTP starts to use the non-primary path for initial data transmission. This is because the recommended value for Path.Max.Retrans in the [RFC4960] is 5, which requires 6 consecutive timeouts before the failover takes place. Before SCTP switches to the non-primary address, SCTP keeps trying to send packets to the primary address and only retransmitted packets are sent to the non-primary address and thus can be received by the receiver. This slow failover process can cause significant performance degradation and is not acceptable in some situations.

Another issue with RFC4960 failover and switchback operation is that once the primary path becomes active again, the traffic is unconditionally switched back to use this path. This is not optimal in some situations. This is further discussed in Section 4.3.

4. SCTP with Potentially-Failed Destination State (SCTP-PF)

To address the issues described in Section 3, this document extends SCTP path management scheme by adding the Potentially Failed state and associated protocol operation. The algorithm is called SCTP Potentially Failed algorithm. SCTP-PF for short. The resulting SCTP path management operation is called SCTP Potentially Failed operation.

4.1. SCTP-PF Concept

The introduction of the Potentially Failed state stems from the following two observations about SCTP’s failure detection procedure:

- To minimize the performance impact during failover, the sender should avoid transmitting data to the failed destination address as early as possible. In the current SCTP path management scheme, the sender stops transmitting data to a destination address only after the destination address is marked Failed (inactive). Thus, a smaller PMR value is better because the sender can transition a destination address to the Failed (inactive) state quicker.

- Smaller PMR values increase the chances of spurious failure detection where the sender incorrectly marks a destination address as Failed (inactive) during periods of temporary congestion. As
[RFC4960] recommends for a coupling of the PMR value and the protocol parameter Association.Max.Retrans (AMR) value such spurious failure detection risks to carry over to spurious association failure detection and closure. Larger PMR values are preferable to avoid spurious failure detection.

From the above observations it is clear that tuning the PMR value involves the following trade off -- a lower value improves performance but increases the chances of spurious failure detection, whereas a higher value degrades performance and reduces spurious failure detection in a wide range of path conditions. Thus, tuning the association’s PMR value is an incomplete solution to address the performance impact during failure.

SCTP-PF defined in this document introduces the new Potentially Failed (PF) destination address state in SCTP’s path management procedure. The new Potentially Failed (PF) destination address state applies to SCTP single-homed operation as well as to SCTP multi-homed operation. The PF state was originally proposed to improve CMT performance [NATARAJAN09]. The PF state is an intermediate state between the Active and Failed states. SCTP’s failure detection procedure is modified to include the PF state. The new failure detection algorithm assumes that loss detected by a timeout implies either severe congestion or failure en-route. After a number of consecutive timeouts on a path, the sender is unsure, and marks the corresponding destination address as in the PF state. A PF destination address is not used for data transmission except when it is the only destination address available (e.g., for single-homed SCTP) or in other special cases (discussed below). The new failure detection algorithm requires only sender-side changes.

4.2. Specification of the SCTP-PF Algorithm

The SCTP-PF operation is specified as follows:

1. The sender maintains a new tunable parameter called PotentiallyFailed.Max.Retrans (PFMR). The RECOMMENDED value of PFMR is 0 when SCTP-PF is used. The PFMR defines a new intermediate PF threshold on the destination address error counter at exceed of which the destination address is classified as PF and related PF state actions are to be taken. By standard RFC4960 semantics a destination address is classified as Inactive once the error counter exceeds PMR. Setting PFMR larger to or equal to PMR does not result in definition of a PF threshold for the destination address. I.e., PFMR set larger to or equal to PMR means that the destination address never will be classified as PF.
2. The error counter of an active destination address is incremented as specified in [RFC4960]. This means that the error counter of the destination address 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. The PFMR threshold defines the point the destination address no longer is considered a good candidate for data transmission and 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 failover to deploy this destination address for data transmission.

   ii. When retransmitting data that has timed out and the sender thus by [RFC4960], section 6.4.1, should attempt to pick a new destination address for data retransmission, 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 address in active state the following considerations are given:

A. An SCTP sender should comply with [RFC4960], section 6.4.1, principles 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.

B. A SCTP-PF sender MAY choose to send data to a destination address in PF state, even if destination addresses in active state exist, have the SCTP-PF sender other means of information available that disqualifies the destination
address in active state from being preferred. However, the
discussion of such mechanisms is outside of the scope of the
SCTP_PF operation specified in 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 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 PF destinations with same error
count, the sender should let the choice among the multiple
PF destination 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.

C. A sender MAY choose to deploy other strategies than the
above when choosing among multiple PF destinations have the
SCTP-PF sender other means of information available that
qualifies a particular destination address for being used.
The SCTP-PF protocol operation specified in this document
makes no assumption of the existence of such other means of
information and specifies for the above as the default
operation of an SCTP-PF sender.

The sender MUST NOT change the state and the error counter of
any destination address regardless of whether it has been chosen
for transmission or not.

5. HEARTBEAT chunks MUST be send to PF destination addresses
regardless of whether the Path Heartbeat function (Section 8.3
of [RFC4960]) is enabled for the destination address or not.
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. The HEARTBEAT sending begins upon that 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 back 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 receipt of a SACK of or a T3-rtx timer expiration on the outstanding data can provide equivalent information. Likewise the timeout of a HEARTBEAT chunk MAY be ignored if data is outstanding towards the destination address.

6. When the sender receives a HEARTBEAT ACK from a destination address in PF state, the sender MUST clear the error counter of the destination address and transition the destination address back to active state. When the sender resumes data transmission on the destination address, it MUST do this following the prescriptions of Section 7.2 of [RFC4960].

7. 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 frequency as described in Section 8.3 of [RFC4960] (when this function is enabled for the destination address).

8. 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) MUST NOT clear the error count for an inactive destination address and MUST NOT transition a PF destination address back to active state, since a sender cannot disambiguate whether the ACK was for the original transmission or the retransmission(s). The same ambiguity concerns the related congestion window growth. The bytes of a newly acknowledged chunk which has been transmitted to multiple destination addresses SHOULD be considered for contribution to the congestion window growth towards the destination address where the chunk was last sent. The contribution of the ACKed bytes to the window growth is subject to the prescriptions described in Section 7.2 of [RFC4960] is fulfilled. A SCTP sender MAY apply a different approach for both the error count handling and the congestion
control growth handling based on unequivocally information on which destination (including multiple destination addresses) the chunk reached. This document makes no reference to what such unequivocally information could consist of, neither how such unequivocally information could be obtained. The design of such an alternative approach is left to implementations.

9. Acknowledgments for chunks that has been transmitted to one destination address only MUST clear the error counter for the destination address and MUST transition a PF destination address 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.

10. The SCTP stack SHOULD provide the ULP with the means to expose the PF state of its destinations as well as the means to notify the state transitions from Active to PF, and vice-versa. When doing this, such an SCTP stack MUST provide the ULP with the means to suppress exposure of PF state and associated state transitions as well.

4.2.1. 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]. One could use higher values of
PMR, which makes the dormant state situations less likely to happen. The downside of increasing the PMR value is that destination address failure detections and notifications of such events to ULP is weakened.

A design goal of SCTP-PF is that it should 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. For this reason, SCTP-PF SHOULD perform the following operations during dormant state, while this is an implementation decision in [RFC4960].

a. When the destination addresses are all in inactive state, the sender MUST choose one destination when data is transmitted. The sender MUST NOT change the state and the error counter of any destination address regardless of whether it has been chosen for transmission or not.

b. 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 such to be an order of magnitude higher than the PMR value. A sender MAY choose to deploy other strategies that the strategy defined by 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.

An SCTP-PF implementation MAY keep the operation during dormant state an implementation decision, but it should be careful not to compromise the fault tolerance of the SCTP operation.

The above prescriptions for SCTP-PF dormant state handling SHOULD NOT be coupled to the value of the PFMR, but solely to the activation of SCTP-PF logic in an SCTP implementation. It is further noted that also a standard [RFC4960] SCTP implementation can use this mode of
operation to improve the fault tolerance (which some implementations already do).

4.3. Permanent Failover

This section describes an OPTIONAL switchback feature called Permanent Failover which is beneficiary to deploy in certain situations.

4.3.1. Background

In [RFC4960], an SCTP sender migrates the traffic back to the original primary destination address once this address becomes active again. As the CWND towards the original primary destination address has to be rebuilt once data transfer resumes, the switch back to use the original primary address is not always optimal. Indeed [CARO02] shows that the switch back to the original primary may degrade SCTP performance compared to continuing data transmission on the same path, especially, but not only, in scenarios where this path’s characteristics are better. In order to mitigate this performance degradation, the Permanent Failover operation was proposed in [CARO02]. When SCTP changes the destination address due to failover, Permanent Failover operation allows SCTP sender to continue data transmission on the new working path even when the old primary destination address becomes active again. This is achieved by having SCTP perform a switch over of the primary path to the alternative working path rather than having SCTP switch back data transfer to the (previous) primary path.

The manner of switch over 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 it depends 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, it is recommended for SCTP to support also, as alternative behavior, the Permanent Failover switch over modes of operation.

4.3.2. Permanent Failover Algorithm

The Permanent Failover operation requires only sender side changes. The details are:
1. The sender maintains a new tunable parameter, called Primary.Switchover.Max.Retrans (PSMR). The PSMR MUST be set greater or equal to the PFMR 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. The recommended value of PSMR is PFMR when Permanent Failover is used. This means that no forced switchback to a previously failed primary path is performed. An implementation of Permanent Failover MUST support the setting of PSMR = PFMR. An implementation of Permanent Failover MAY support setting of PSMR > PFMR.

5. It MUST be possible to disable the Permanent Failover and obtain the standard switchback operation of [RFC4960].

To support optimal operation in a wider range of network scenarios, it is proposed for an SCTP-PF implementation to implement Permanent Failover operation as an optional feature. The implementation of the Permanent Failover feature is optional for an SCTP-PF implementation. For an SCTP implementation that implements Permanent Failover, this specification RECOMMENDS that the standard RFC4960 switchback operation is retained as the default operation.

5. 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.

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 FFMR and PSMR parameters described in Section 4. 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().

5.1. 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.

5.2. 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 and before the primary path is changed automatically. 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 wildcard 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. Setting of spt_pathcpthld < spt_pathpfthld should be rejected with errno set to EINVAL. An implementation MAY support only setting of spt_pathcpthld = spt_pathpfthld and spt_pathcpthld = 0xffff. In this case it shall reject setting of other values with errno set to EINVAL.

5.3. 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 5.1. 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:

```c
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.

6. Security Considerations

Security considerations for the use of SCTP and its APIs are discussed in [RFC4960] and [RFC6458]. The logic described here is for sender-side only enabled by configuration and does not have any
impacts on protocol messages on the wire. No new chunk type or new field parameter is not required in this document.

7. IANA Considerations

This document does not create any new registries or modify the rules for any existing registries managed by IANA.

8. 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 Telco signaling environments for several years with good results.

9. References

9.1. Normative References


9.2. Informative References


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. In fact, this is recommended in some research results [JUNGMAIER02] [GRINNEMO04] [FALLON08]. For example, if when Path.Max.Retrans=0, SCTP switches to another destination address on a single timeout. This smaller value for Path.Max.Retrans can results in spurious failover, which might be a problem.
Unlike SCTP-PF, the interval for heartbeat packets is governed by ‘HB.interval’ even during failover process. ‘HB.interval’ is usually set in the order of seconds (recommended value is 30 seconds). When the primary path becomes inactive, the next HEARTBEAT can be transmitted only seconds later. Meanwhile, the primary path may have recovered. In such situations, post failover, an endpoint is forced to wait on the order of seconds before the endpoint can resume transmission on the primary path. However, using smaller value for ‘HB.interval’ might help this situation, but it will be the waste of bandwidth in most cases.

In addition, smaller Path.Max.Retrans values also affect ‘Association.Max.Retrans’ values. When the SCTP association’s error count (sum of error counts on all ACTIVE paths) exceeds Association.Max.Retrans threshold, the SCTP sender considers the peer endpoint unreachable and terminates the association. Therefore, 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. To avoid such inconsistent behavior an SCTP implementation SHOULD reduce Association.Max.Retrans accordingly whenever it reduces Path.Max.Retrans. However, smaller Association.Max.Retrans value increases 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 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.
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
Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol

draft-ietf-tsvwg-sctp-ndata-03.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a message oriented transport protocol supporting arbitrary large user messages. However, the sender can not interleave different user messages which causes head of line blocking at the sender side. To overcome this limitation, this document adds a new data chunk to SCTP.

Whenever an SCTP sender is allowed to send a user data, it can possibly choose from multiple outgoing SCTP streams. Multiple ways for this selection, called stream schedulers, are defined. Some of them don’t require the support of user message interleaving, some do.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2015.
Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction ............................................. 3
   1.1. Overview ............................................ 3
   1.2. Conventions ......................................... 4
2. User Message Interleaving ................................. 4
   2.1. The I-DATA Chunk supporting User Message Interleaving . 4
   2.2. Procedures ........................................... 5
       2.2.1. Negotiation ..................................... 5
       2.2.2. Sender Side Considerations .................... 6
       2.2.3. Receiver Side Considerations ................... 6
   2.3. Interaction with other SCTP Extensions ................. 6
       2.3.1. SCTP Partial Reliability Extension ............ 7
       2.3.2. SCTP Stream Reconfiguration Extension ........ 7
3. Stream Schedulers .......................................... 7
   3.1. Stream Scheduler without User Message Interleaving Support ............................................. 7
   3.2. Stream Scheduler with User Message Interleaving Support .................................................. 7
4. Socket API Considerations ................................ 7
   4.1. SCTP_ASSOC_CHANGE Notification ...................... 7
   4.2. Socket Options ....................................... 7
       4.2.1. Enable or Disable the Support of User Message Interleaving ........................................... 8
       4.2.2. Get or Set the Stream Scheduler (SCTP_PLUGGABLE_SS) .................................................. 9
       4.2.3. Get or Set the Stream Scheduler Parameter (SCTP_SS_VALUE) .......................................... 10
5. IANA Considerations ...................................... 11
6. Security Considerations .................................. 12
7. Acknowledgments .......................................... 12
8. References .............................................. 12
   8.1. Normative References ................................ 12
   8.2. Informative References ............................. 13
Authors’ Addresses ........................................ 13
1. Introduction

1.1. Overview

When SCTP [RFC4960] was initially designed it was mainly envisioned for 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 MTU sized message would be too small. Unfortunately this design decision, though valid at the time, did not account for other applications which might send very large messages over SCTP. When such large messages are now sent over SCTP a form of sender side head of line blocking becomes created within the protocol. This head of line blocking is caused by the use of the Transmission Sequence Number (TSN) for two 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.

The protocol requires all fragments of a user message to have consecutive TSNs. Therefore it is impossible for the sender to interleave different user messages.

This document describes a new Data chunk called I-DATA. This chunk incorporates all the flags and fields except the Stream Sequence Number (SSN) and properties of the current SCTP Data chunk but also adds two new fields in its chunk header, the Fragment Sequence Number (FSN) and the Message Identifier (MID). Then the FSN is only used for reassembling all fragments with the same MID and the TSN only for the reliability. The MID is also used for ensuring ordered delivery, therefore replacing the stream sequence number. Therefore, the head of line blocking caused by the original design is avoided.

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 and no DATA chunks. It should be noted, that an SCTP implementation needs to support the coexistence of associations using DATA chunks and associations using I-DATA chunks.

This document also defines several stream schedulers for general SCTP associations. If I-DATA support has been negotiated, several more schedulers are available.
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

2.1. The I-DATA Chunk supporting User Message Interleaving

The following Figure 1 shows the new I-DATA chunk allowing user messages 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              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              TSN                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Stream Identifier      |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Message Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Payload Protocol Identifier / Fragment Sequence Number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                   \                           /
                                   \             User Data               /
                                   \                           /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: I-DATA chunk format

The only differences between the I-DATA chunk in Figure 1 and the DATA chunk defined in [RFC4960] and [RFC7053] is the addition of the new Message Identifier (MID) and Fragment Sequence Number (FSN) and the removal of the Stream Sequence Number (SSN). However, the lower 16-bit of the MID can be used as the SSN if necessary. 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].

Reserved: 16 bits (unsigned integer)

This field is reserved. It SHOULD be set to 0 by the sender and 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 two counters, one for ordered messages, one for unordered messages, for each outgoing streams. All 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. For ordered user messages, the lower 16 bit of the MID can be used as a SSN if required. 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. 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 arbitrary 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. 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.

2.2.1. Negotiation

A sender MUST NOT send a I-DATA chunk unless both peers have indicated its support of the I-DATA chunk type within the Supported Extensions Parameter as defined in [RFC5061]. If I-DATA support has been negotiated on an association, I-DATA chunks MUST be used for all user messages and DATA-chunk MUST NOT be used. If I-DATA support has not been negotiated on an association, DATA chunks MUST be used for all user messages and I-DATA chunks MUST NOT be used.

A sender MUST NOT use the I-DATA chunk unless the user has requested that use (e.g. via the socket API, see Section 4). This constraint is made since usage of this chunk requires that the application be
willing to interleave messages upon reception within an association. This is not the default choice within the socket API (see [RFC6458]) thus the user MUST indicate support to the protocol of the reception of completely interleaved messages. Note that for stacks that do not implement [RFC6458] they may use other methods to indicate interleaved message support and thus enable the usage of the I-DATA chunk, the key is that the the stack MUST know the application has indicated its choice in wanting to use the extension.

2.2.2. Sender Side Considerations

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’ 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 should also imply late binding of the actual TSN to any chunk being sent. This way other messages from other streams may be interleaved with the fragmented message.

The sender MUST NOT have more than one ordered fragmented message being produced in any one stream. The sender MUST NOT have more than one un-ordered fragmented message being produced in any one stream. The sender MAY have one ordered and one unordered fragmented message being produced within a single stream. At any time multiple streams MAY be producing an ordered or unordered fragmented message.

2.2.3. Receiver Side Considerations

Upon reception of an SCTP packet containing a I-DATA chunk if the message needs to be reassembled, then the receiver MUST use the FSN for reassembly of the message and not the TSN. Note that a non-fragmented messages is indicated by the fact that both the ‘E’ and ‘B’ bits are set. An ordered or unordered fragmented message is thus identified with any message not having both bits set.

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, the lower 16 bits of the MID counters for ordered messages MUST be used when filling the SSNs in the FORWARD-TSN chunk.

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 MID MUST be reset to 0 correspondingly.

3. Stream Schedulers

3.1. Stream Scheduler without User Message Interleaving Support

TBD.

3.2. Stream Scheduler with User Message Interleaving Support

TBD.

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. SCTP_ASSOC_CHANGE Notification

When an SCTP_ASSOC_CHANGE notification 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 listen in the sac_info field.

4.2. Socket Options
### 4.2.1. Enable or Disable the Support of User Message Interleaving

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.

User message interleaving is disabled per default.

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]:

```c
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. This allows the reception from multiple streams simultaneously. Failure to set this option can possibly lead to application deadlock.
4.2.2. Get or Set the Stream Scheduler (SCTP_PLUGGABLE_SS)

A stream scheduler can be selected with the SCTP_PLUGGABLE_SS 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]:

```c
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_ROUND_ROBIN or SCTP_SS_FIRST_COME.

- **SCTP_SS_ROUND_ROBIN**: This scheduler provides a fair scheduling based on the number of user messages by cycling around non-empty stream queues.

- **SCTP_SS_ROUND_ROBIN_PACKET**: This is a round-robin scheduler but only bundles user messages of the same stream in one packet. This minimizes head-of-line blocking when a packet is lost because only a single stream is affected.

- **SCTP_SS_PRIORITY**: Scheduling with different priorities is used. Streams having a higher priority will be scheduled first and when multiple streams have the same priority, the default scheduling should be used for them. 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_FAIR_BANDWITH**: A fair bandwidth distribution between the streams can be activated using this value. This scheduler considers the lengths of the messages of each stream and
schedules them in a certain way to maintain an equal bandwidth for all streams.

**SCTP_SS_WEIGHTED_ROUND_ROBIN:** A weighted round robin distribution between the streams can be activated using this value. This scheduler considers the lengths of the messages of each stream and schedules them in a certain way to use the bandwidth according to the given weights.

**SCTP_SS_FIRST_COME:** The simple first-come, first-serve algorithm is selected by using this value. 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.

4.2.3. Get or Set the Stream Scheduler Parameter (SCTP_SS_VALUE)

Some schedulers require additional information to be set for single streams as shown in the following table:

<table>
<thead>
<tr>
<th>name</th>
<th>per stream info</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCTP_SS_DEFAULT</td>
<td>no</td>
</tr>
<tr>
<td>SCTP_SS_RR</td>
<td>no</td>
</tr>
<tr>
<td>SCTP_SS_RR_INTER</td>
<td>no</td>
</tr>
<tr>
<td>SCTP_SS_RR_PKT</td>
<td>no</td>
</tr>
<tr>
<td>SCTP_SS_RR_PKT_INTER</td>
<td>no</td>
</tr>
<tr>
<td>SCTP_SS_PRIO</td>
<td>yes</td>
</tr>
<tr>
<td>SCTP_SS_PRIO_INTER</td>
<td>yes</td>
</tr>
<tr>
<td>SCTP_SS_FB</td>
<td>no</td>
</tr>
<tr>
<td>SCTP_SS_FB_INTER</td>
<td>no</td>
</tr>
<tr>
<td>SCTP_SS_WRR</td>
<td>yes</td>
</tr>
<tr>
<td>SCTP_SS_WRR_INTER</td>
<td>yes</td>
</tr>
<tr>
<td>SCTP_SS_FCFS</td>
<td>no</td>
</tr>
</tbody>
</table>

This is achieved with the SCTP_SS_VALUE option and the corresponding struct sctp_stream_value. The definition of struct sctp_stream_value is as follows:

```c
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 for 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.

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 type 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.

A new chunk type has to be assigned by IANA. IANA should assign this value 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       | New DATA chunk (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:
<table>
<thead>
<tr>
<th>Chunk Flag Value</th>
<th>Chunk Flag Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>E bit</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>0x02</td>
<td>B bit</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>0x04</td>
<td>U bit</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>0x08</td>
<td>I bit</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>0x10</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0x40</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0x80</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>

6. Security Considerations

This document does not add any additional security considerations in addition to the ones given in [RFC4960] and [RFC6458].

7. Acknowledgments

The authors wish to thank Christer Holmberg, Karen E. Egede Nielsen, Irene Ruengeler, and Lixia Zhang for her invaluable comments.

8. References

8.1. Normative References


8.2. Informative References


Authors’ Addresses

Randall R. Stewart
Netflix, Inc.
Chapin, SC  29036
US

Email: randall@lakerest.net

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

Email: tuexen@fh-muenster.de

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvás  02420
FI

Email: Salvatore.Loreto@ericsson.com
Additional Policies for the Partial Reliability Extension of the Stream Control Transmission Protocol

draft-ietf-tsvwg-sctp-prpolicies-07.txt

Abstract

This document defines two additional policies for the Partial Reliability Extension of the Stream Control Transmission Protocol (PR-SCTP) allowing to limit the number of retransmissions or to prioritize user messages for more efficient send buffer usage.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 11, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
Internet-Draft Additional PR-SCTP Policies February 2015

carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 2
2. Conventions ................................................... 3
3. Additional PR-SCTP Policies ................................. 3
   3.1. Limited Retransmissions Policy ......................... 3
   3.2. Priority Policy ....................................... 3
4. Socket API Considerations ..................................... 4
   4.1. Data Types ............................................ 4
   4.2. Support for Added PR-SCTP Policies .................... 4
   4.3. Socket Option for Getting the Stream Specific PR-SCTP
        Status (SCTP_PR_STREAM_STATUS) .................... 5
   4.4. Socket Option for Getting the Association Specific PR-
        SCTP Status (SCTP_PR_ASSOC_STATUS) ............... 6
   4.5. Socket Option for Getting and Setting the PR-SCTP Support
        (SCTP_PR_SUPPORTED) .................................. 7
5. IANA Considerations ............................................ 8
6. Security Considerations ....................................... 8
7. Acknowledgments ................................................ 8
8. References ...................................................... 8
   8.1. Normative References ................................... 8
   8.2. Informative References ................................. 9
Authors’ Addresses ................................................. 9

1. Introduction

The SCTP Partial Reliability Extension (PR-SCTP) defined in [RFC3758]
provides a generic method for senders to abandon user messages. The
decision to abandon a user message is sender side only and the exact
condition is called a PR-SCTP policy ([RFC3758] refers to them as
‘PR-SCTP Services’). [RFC3758] also defines one particular PR-SCTP
policy, called Timed Reliability. This allows the sender to specify
a timeout for a user message after which the SCTP stack abandons the
user message.

This document specifies the following two additional PR-SCTP
policies:

Limited Retransmission Policy: Allows to limit the number of
retransmissions.
Priority Policy: Allows to discard lower priority messages if space for higher priority messages is needed in the send buffer.

2. Conventions

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

3. Additional PR-SCTP Policies

This section defines two new PR-SCTP policies, one in each subsection.

Please note that it is REQUIRED to implement [RFC3758], if you want to implement these additional policies. However, these additional policies are OPTIONAL when implementing [RFC3758].

3.1. Limited Retransmissions Policy

Using the Limited Retransmission Policy allows the sender of a user message to specify an upper limit for the number of retransmissions for each DATA chunk of the given user messages. The sender MUST abandon a user message if the number of retransmissions of any of the DATA chunks of the user message would exceed the provided limit. The sender MUST perform all other actions required for processing the retransmission event, such as adapting the congestion window and the retransmission timeout. Please note that the number of retransmissions includes both fast and timer-based retransmissions.

The sender MAY limit the number of retransmissions to 0. This will result in abandoning the message when it would get retransmitted for the first time. The use of this setting provides a service similar to UDP, which also does not perform any retransmissions.

Please note that using this policy does not affect the handling of the thresholds 'Association.Max.Retrans' and 'Path.Max.Retrans' as specified in Section 8 of [RFC4960].

The WebRTC protocol stack (see [I-D.ietf-rtcweb-data-channel]), is an example of where the Limited Retransmissions Policy is used.

3.2. Priority Policy

Using the Priority Policy allows the sender of a user message to specify a priority. When storing a user message in the send buffer while there is not enough available space, the SCTP stack at the sender side MAY abandon other user message(s) of the same SCTP
association (with the same or a different stream) with a priority lower than the provided one. User messages sent reliable are considered having a priority higher than all messages sent with the Priority Policy. The algorithm for selecting the message(s) being abandoned is implementation specific.

After lower priority messages have been abandoned high priority messages can be transferred without the send call blocking (if used in blocking mode) or the send call failing (if used in non-blocking mode).

The IPFIX protocol stack (see [RFC7011]) is an example of where the Priority Policy can be used. Template records would be sent with full reliability, while billing, security-related, and other monitoring flow records would be sent using the Priority Policy with varying priority. The priority of security related flow-records would be chosen higher than the the priority of monitoring flow records.

4. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to support the newly defined PR-SCTP policies, to provide some statistical information and to control the negotiation of the PR-SCTP extension during the SCTP association setup.

Please note that this section is informational only.

4.1. Data Types

This section uses data types from [IEEE.1003-1G.1997]: uintN_t means an unsigned integer of exactly N bits (e.g. uint16_t). This is the same as in [RFC6458].

4.2. Support for Added PR-SCTP Policies

As defined in [RFC6458], the PR-SCTP policy is specified and configured by using the following sctp_prinfo structure:

```c
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:

<table>
<thead>
<tr>
<th>pr_policy</th>
<th>pr_value</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCTP_PR_SCTP_NONE</td>
<td>Ignored</td>
<td>[RFC6458]</td>
</tr>
<tr>
<td>SCTP_PR_SCTP_TTL</td>
<td>Lifetime in ms</td>
<td>[RFC6458]</td>
</tr>
<tr>
<td>SCTP_PR_SCTP_RTX</td>
<td>Number of retransmissions</td>
<td>Section 3.1</td>
</tr>
<tr>
<td>SCTP_PR_SCTP_PRIO</td>
<td>Priority</td>
<td>Section 3.2</td>
</tr>
</tbody>
</table>

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:

```c
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:

```c
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 Romascun, Irene Ruengeler, Jamal Hadi Salim, Joseph Salowey, Brian Trammell, and Vlad Yasevich for their invaluable comments.

8. References

8.1. Normative References

8.2. Informative References


Authors’ Addresses

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

Email: tuexen@fh-muenster.de
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.
# Table of Contents

1. Design Team Charter ............................................. 4  
2. Overview ............................................................ 4  
3. Common Issues ..................................................... 6  
4. Scope ................................................................. 6  
5. Assumptions ......................................................... 7  
6. Terminology .......................................................... 7  
7. Entropy ............................................................... 7  
8. Next-protocol indication .......................................... 9  
9. MTU and Fragmentation ........................................... 10  
10. OAM ................................................................. 10  
11. Security Considerations .......................................... 12  
  11.1. Encapsulation-specific considerations ..................... 12  
  11.2. Virtual network isolation ................................... 14  
  11.3. Packet level security ....................................... 15  
12. QoS ................................................................. 15  
13. Congestion Considerations ....................................... 16  
14. Header Protection .................................................. 18  
15. Extensibility Considerations .................................... 19  
  15.1. Next-protocol ................................................ 22  
16. Layering Considerations .......................................... 22  
17. Service model ...................................................... 23  
18. Hardware Friendly ................................................ 24  
  18.1. Considerations for NIC offload ............................. 26  
19. Middlebox Considerations ....................................... 29  
20. Related Work ....................................................... 30  
21. Acknowledgements ............................................... 31  
22. Open Issues ....................................................... 31  
23. References ........................................................ 32  
  23.1. Normative References ....................................... 32  
  23.2. Informative References .................................... 33  
Authors’ Addresses .................................................. 36

1. Design Team Charter

There have been multiple efforts over the years that have resulted in new or modified data plane behaviors involving encapsulations. That includes IETF efforts like MPLS, LISP, and TRILL but also industry efforts like VXLAN and NVGRE. These collectively can be seen as a source of insight into the properties that data planes need to meet. The IETF is currently working on potentially new encapsulations in NVO3 and SFC and considering working on BIER. In addition there is work on tunneling in the INT area.

This is a short term design team chartered to collect and construct useful advice to parties working on new or modified data plane behaviors that include additional encapsulations. The goal is for the group to document useful advice gathered from interacting with ongoing efforts. An Internet Draft will be produced for IETF92 to capture that advice, which will be discussed in RTGWG.

Data plane encapsulations face a set of common issues such as:
- How to provide entropy for ECMP
- Issues around packet size and fragmentation/reassembly
- OAM - what support is needed in an encapsulation format?
- Security and privacy
- QoS
- Congestion Considerations
- IPv6 header protection (zero UDP checksum over IPv6 issue)
- Extensibility - e.g., for evolving OAM, security, and/or congestion control
- Layering of multiple encapsulations e.g., SFC over NVO3 over BIER

The design team will provide advice on those issues. The intention is that even where we have different encapsulations for different purposes carrying different information, each such encapsulation doesn’t have to reinvent the wheel for the above common issues.

The design team will look across the routing area in particular at SFC, NVO3 and BIER. It will not be involved in comparing or analyzing any particular encapsulation formats proposed in those WGs and BoFs but instead focus on common advice.

2. Overview

The references provide background information on NVO3, SFC, and BIER. In particular, NVO3 is introduced in [RFC7364], [RFC7365], and [I-D.ietf-nvo3-arch]. SFC is introduced in [I-D.ietf-sfc-architecture] and [I-D.ietf-sfc-problem-statement]. Finally, the information on BIER is in [I-D.shepherd-bier-problem-statement],

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.

- **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.
- **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 each hop in the service chain even if the meta-data is unmodified.
- **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
- **NVO3** might need some extensions for OAM and security.
- **SFC** is all about carrying service meta-data along a path, and different services might need different types and amount of meta-data.
- **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

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.
- 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:
- Design some new encapsulation header to rule them all
- Design yet another new NVO3 encapsulation header
- Try to select the best encapsulation header
- Evaluate any existing and proposed encapsulations

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 environments, it is reasonable to make these assumptions:
- 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.)
- 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 encaps 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 encap 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 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.]

8. Next-protocol indication

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 was as IPv4 being 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.

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:

- Reuse Ethernet types - makes it easy to carry existing L2 and L3 protocols
- Reuse IP protocol numbers - makes it easy to carry e.g., ESP but brings in all existing protocol numbers many of which would never be used directly on top of the encapsulation protocol.
- 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.

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.

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. 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 encaps and decaps nodes. [Aerolink is one example of such a protocol.]

Depending on the payload carried by the encapsulation there are some additional possibilities:
- If payload is IPv4/6 then the underlay path MTU could be used to report end-to-end path MTU.
- 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.

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:
- 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.
- 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 encaps can choose between one of:
- 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
- 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.) OAM bit in encaps must in any case be excluded from the entropy.

There can be several ways to prevent OAM packets from accidentally being forwarded to hosts using:
- A bit in the frame (as in TRILL) indicating OAM
- 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.

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 encaps the same way ICMP is associated with IP. Would it make sense for the IETF to develop a common Encaps 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.

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 at least uniform security mechanisms for encapsulation.

There are at least three considerations for security:
- Anti-spoofing/virtual network isolation
- Interaction with packet level security such as IPsec
- 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.
- 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 at least a statement on security
    * Tenant may implement their own additional layers of security
    * Regulation and certification considerations

  o Trend towards tighter security:
    * Tenants’ data in network increases in volume and value, attacks become more sophisticated
    * Large DCs already encrypt everything on disk
* DCs likely to encrypt inter-DC traffic at this point, use TLS to Internet.
* Encryption within DC is becoming more commonplace, becomes ubiquitous when cost is low enough.
* Cost/performance considerations. Cost of support for strong security has made strong network security in DCs prohibitive.
* Are there lessons from MacSec?

11.2. Virtual network isolation

The first requirement is isolation between virtual networks. Packets sent in one virtual network should never be illegitimately received by a node in another virtual network. Isolation should be protected in the presence of malicious attacks or inadvertent packet corruption.

The second requirement is sender authentication. Sender identity is authenticated to prevent anti-spoofing. Even if an attacker has access to the packets in the network, they cannot send packets into a virtual network. This may have two possibilities:

- Pairwise sender authentication. Any two communicating hosts negotiate a shared key.
- Group authentication. A group of hosts share a key (this may be more appropriate for multicast of encapsulation).

Possible security solutions:

- 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 \( F(\text{cookie}, \text{VNI ID}, \text{source addr}) \). 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.

- 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 \( H(\text{key}, \text{VNI ID}, \text{addrs}, \text{salt}) \). 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 encaps 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 and visible to the network (for instance in nvo3 we may want to firewall based on VNI ID even if packet is encrypted). In this model protocol stack may be something like IP|UDP|Encap|ESP|IP in tunnel mode, but there’s nothing that prevents using transport mode (this looks a lot like ESP/UDP [RFC3948]). The encapsulation and security are probably done together at encapsulator and resolved at decapsulator. Since the encapsulation header is outside of the security coverage, this may itself require security also (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). In this case of security applied to encap payload, this does present a bit of protocol layer inversion in the header (encapsulation refers to overlay, but ESP operates on underlay), but this should be okay as long as semantics are clear and processing is deterministic.

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].

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 encaps 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.

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".

For this reason, where an encaps 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 [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.wai-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.

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.

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).

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.

- 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.

- 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.

- 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.

- 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 encaps header would need to be able to know which extensions can be changed in flight.

15.1. Next-protocol

[Note that there is editorial duplication between this section and the Next Protocol Indication section earlier in the document]

Choices:

- Payload type implied (by UDP port for instance). ESP, L2TP, MPLS, over UDP are example, Linux Foo-Over-UDP is example implementation. This model is simple, however allocating a port for every possible protocol might be difficult given the trend towards port conservation as described in [I-D.ietf-tsvwg-port-use].

- Encapsulation contains a next header field. Possibilities:
  - EtherType: GRE protocol field is example. Allows encapsulation of any EtherType (including IPv4, IPv6, end Ethernet). Disadvantages are that it is 16 bits for less than 100 needed values, and the number space is controlled by the IEEE 802 RAC.
  - IP protocol: IPv6 extension headers, ESP are examples. Allows encapsulation of any IP protocol including Ethernet via ETHERIP, IPv4, IPv6, IPsec protocols, UDP (transport mode considerations needed). IANA managed eight bit values, presumably more difficult to get an assigned number than to get a transport port assignment.
  - Protocol specific number space. Example PPP. Could be 8 or 16 bits and would be IANA controlled. Primary advantage is that it might be easier to define protocols for encapsulation that are not defined in other number spaces (802.11 for instance). Disadvantage is that it represents yet another number space to be managed and doesn’t leverage existing ones.

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.

- 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 encaps header or inner headers and payload. And a BIER router is allowed to modify the BIER header.
- Alternatively such a device can become visible at a higher layer. E.g., a middlebox could become an decaps + function + encaps which means it will generate a new encaps header.

The design team asked itself some additional questions:

- Would it make sense to have a common encaps 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.
- 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 encaps 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 encapss. Potentially the tunnel control points in the form of an ingress and egress will become a platform for fixing suboptimal behavior elsewhere in the network. For example, this document suggests that some congestion handling might be needed to handle non-congestion controlled traffic that gets tunneled, and also that fairness/QoS policing can be deployed on those devices. Others have suggested that tunnels is one way to deploy ECN without having to add ECN support in the endpoints [I-D.briscoe-conex-data-centre].

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 Separation of NVO3 header from SFC header such that an encap can also be processed by forwarding hardware (who can only process network virtualization and pass the service chaining function to another device specialized in service offering)
  o Avoid full packet checksums in the encapsulation if possible. 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). Hardware can handle optional information as long as when the information is present it is found in one and only one place in the header. Typical TLV encoding of options does not have that property since
the order of TLVs is unconstrained.
  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 (CHECKSUMCOMPLETE 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:

- Split the TCP payload into segments which allow packets with size less than or equal to MTU.
- For each created segment:
  1. Replicate the TCP header and all preceding headers of the original packet.
  2. Set payload length fields in any headers to reflect the length of the segment.
  3. Set TCP sequence number to correctly reflect the offset of the TCP data in the stream.
  4. Recompute and set any checksums that either cover the payload of the packet or cover header which was changed by setting a payload length.

Following this general process, TSO can be extended to support TCP encapsulation 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.

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 encaps-aware middleboxes should avoid unintended consequences in those (and perhaps other) areas.
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 encodings 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].

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 David Black, Andy Malis, and Radia Perlman.

22. Open Issues

- **Middleboxes:**
  - Due to OAM there are constraints on middleboxes in general. If middleboxes inspect the packet past the outer IP+UDP and encaps 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?

- **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.

- **Common OAM error reporting protocol?**

- 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 encaps 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. References

23.1. Normative References


23.2. Informative References

[I-D.briscoe-conex-data-centre]

[I-D.farrelll-mpls-opportunistic-encrypt]

[I-D.gross-geneve]

[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]

[I-D.ietf-nvo3-arch]

[I-D.ietf-sfc-architecture]

[I-D.ietf-sfc-problem-statement]

[I-D.ietf-trill-oam-fm]

[I-D.ietf-tsvwg-circuit-breaker]

[I-D.ietf-tsvwg-gre-in-udp-encap]

[I-D.ietf-tsvwg-port-use]

[I-D.quinn-sfc-nsh]
Quinn, P., Guichard, J., Surendra, S., Smith, M., Henderickx, W., Nadeau, T., Agarwal, P., Manur, R.,

[I-D.saldana-tsvwg-simplemux]

[I-D.shepherd-bier-problem-statement]

[I-D.sridharan-virtualization-nvgre]

[I-D.wei-tsvwg-tunnel-congestion-feedback]

[I-D.wijnands-bier-architecture]

[I-D.wijnands-mpls-bier-encapsulation]

[I-D.xu-bier-encapsulation]

[IEEE802.1Q-2014]
IEEE, "IEEE Standard for Local and metropolitan area
networks—Bridges and Bridged Networks", IEEE Std 802.1Q-2014, 2014,

(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
Requirements and reference architecture for Mobile Throughput Guidance
Exposure
draft-sprecher-mobile-tg-exposure-req-arch-01.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 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."
1. Introduction

The following sub-sections present the problem statement and the solution principles.
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.

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 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.

8. The TCP server MAY reside within the mobile operator’s network (behind the mobile core network) or in the Internet.

9. The TCP server MAY use the mobile throughput guidance information to assist TCP.

10. 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.

11. The Throughput Guidance Provider SHOULD provide the mobile throughput guidance information periodically, starting from the initiation of the flow.

12. The frequency (in milliseconds) at which mobile throughput guidance needs to be exposed SHALL be configurable.

13. 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.

14. 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.

15. 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.

16. 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.
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:

- Determine the size of the initial congestion window
- Determine when to exit the slow start phase
- Determine the size of the congestion window during the congestion avoidance phase
- 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


9.2. Informative References


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

Hannu Flinck
Nokia Networks
Helsinki
FI

Phone: +358504839522
Email: hannu.flinck@nsn.com

Helen Parsons
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: helenparsons@google.com

Peter Cosimini
Vodafone
Vodafone House, The Connection
Newbury, RG14 2FN
UK

Email: peter.cosimini@vodafone.com

Ram Gopal
Nokia Networks
Mountain View
US

Phone: +17815264572
Email: ram.gopal@nsn.com
Tunnel Congestion Feedback

draft-wei-tsvwg-tunnel-congestion-feedback-03

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 router. A basic model for measuring tunnel congestion and feedback is described, and a protocol for carrying the feedback data is outlined.

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.
Table of Contents

1. Introduction ........................................... 3
2. Conventions and Terminology ............................... 4
   2.1 Conventions ........................................ 4
   2.2 Terminology ....................................... 4
3. Problem Statement ........................................ 5
   3.1 3GPP network scenario ............................... 6
   3.2 Network Function Virtualization Scenario .............. 7
   3.3 Data Center Tenancy Scenario ........................ 9
4. Congestion Control Model ................................ 9
   4.1 Congestion Calculation ............................... 10
   4.2 Data Information ................................... 12
   4.3 Congestion Feedback ................................ 12
   4.4 Congestion Control ................................ 13
5. Congestion Feedback Protocol ............................. 13
   5.1 Properties of Candidate Protocol ..................... 13
   5.2 IPFIX Extensions for Congestion Feedback .............. 14
   5.3 Other Protocols ................................... 18
6. Benefits ................................................. 18
7. Security Considerations ................................ 18
8. IANA Considerations .................................... 18
9. References ............................................. 19
   9.1 Normative References ............................... 19
   9.2 Informative References ............................. 19
Authors’ Addresses ........................................... 20
1. Introduction

In current practice of Internet protocol, encapsulation of IP headers is always the technical proposal for overlay networking scenarios. For example, mobile network are designed to encapsulate inner IP header and application layer header chain through IP header, UDP header and GTP-U header. It is also designed to fulfill the mobility, QoS control, bearer management and other specific application of the mobile network. Some organization’s private network encrypt IP header by Internet tunnel solutions with private key or certification approaches to setup VPN (virtual private network) over WAN (wide area network).

Congestion is the situation that traffic input exceeds throughput of any segment of transmission path, which can result from transportation constraints and interface/processor overload. In general, congestion seen as the cause of packet loss or unexpected delay to network end points. End to end congestion protocols (e.g. ECN [RFC 3168] and ECN handling for tunneling scenario [RFC6040]) are discussed in IETF.

In IP header encapsulation cases, IP headers should be carried over transportation protocol like TCP or UDP, which influences the explicit congestion control feedback, since the receiver should mark ECN in TCP acknowledgment. On the other hand, packet loss and performance degradation should not be recognized by network elements, for instance the tunnel ingress and egress entity, when network segment is encapsulated by IP header and UDP header chain. That causes management problem when tunnel segment is considered as an independent administration domain, and network operator intents to keep network operation reliable.

This document describes a mechanism for feedback of congestion observed in IP tunnels usages. Common tunnel deployments such as mobile backhaul networks, VPNs and other IP-in-IP tunnels can be congested as a result of sustained high load.

Network providers use a number of methods to deal with high load conditions including proper network dimensioning, policies for preferential flow treatment and selective offloading among others. The mechanism proposed in this document is expected to complement them and provide congestion information that to allow making better, policies and decisions.

The model and general solution proposed in chapter 4 consist of identifying congestion marks set in the tunnel segment, and feeding back the congestion information from the egress to the ingress of the tunnel. Measuring congestion of a tunnel segment is based on counting
outer packet CE marks for packets that have ECT marks in the inner packet. This proposal depends on statistical marking of congestion and uses the method described in RFC 6040 [RFC6040], Appendix C.

In chapter 5 the desired properties of the congestion information conveying protocol are outlined, and IPFIX [RFC5101] as a candidate protocol for these extensions is explored further.

2. Conventions and Terminology

2.1 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].

2.2 Terminology

Tunnel: A channel over which encapsulated packets traverse across a network.

Encapsulation: The process of adding control information when it passes through the layered model.

Encapsulator: The tunnel endpoint function that adds an outer IP header to tunnel a packet, the encapsulator is considered as the "ingress" of the tunnel.

Decapsulator: The tunnel endpoint function that removes an outer IP header from a tunneled packet, the decapsulator is considered as the "egress" of the tunnel.

Outer header: The header added to encapsulate a tunneled packet.

Inner header: The header encapsulated by the outer header.

E2E: End to End.

VPN: Virtual Private Network is a technology for using the Internet or another intermediate network to connect computers to isolated remote computer networks that would otherwise be inaccessible.

GRE: Generic Routing Encapsulation.

IPFIX: IP Flow Information Export. An IETF protocol to export flow information from routers and other devices.
3. Problem Statement

Network traffic congestion control plays a significant role in network performance management, and sustaining congestion could impact subscriber’s experience. Currently the solution of network congestion problem mainly focuses on end-to-end method, i.e. ECN [RFC3168], and the traffic sender are in charge of reducing traffic rates in case of network congested. But sometimes it’s not always reliable to dependent on end hosts to solve the congestion situation, because some end hosts may not support ECN, or even ECN is supported by end hosts some traffics, e.g. UDP-based traffic, may not support ECN.

Though the congestion happens in operator’s network, in case that the congestion information is transparent to operator, network administration would be hard to take action to control the network traffic of reason to network congestion. To improve the performance of the network, it’s better for operator to take network congestion situation into network traffic management.

Many kinds of tunnels are widely deployed in current networks, even in some scenarios all traffics transmitted through designated tunnel(s).

Because the ingress and egress of tunnel are usually deployed by operator, so it’s easy for operator to execute operator’s policy, for example gating, flow control and dropping. The tunnel feedback mechanism should be feasible for operator to collect network congestion information in encapsulation segment. After obtaining
congestion information, operator could make policy at tunnel ingress for traffic management taking these information into consideration.

ECN handling mechanisms in RFC 6040 specifies how ECN should be handled for tunneling. In addition, RFC 6040, Appendix C provides guidance to calculate congestion experienced in the tunnel itself. However, there is no standardized mechanism by which the congestion information inside the tunnel can be fed back from egress to ingress router.

In the following sub-sections, some network tunnel scenarios are discussed.

3.1 3GPP network scenario

Tunnels, including GRE [RFC2784], GTP [TS29.060], IP-in-IP [RFC2003] or IPSec [RFC4301] etc, are widely deployed in 3GPP networks. And in 3GPP network tunnels are used to carry end user flows within the backhaul network such as shown in Figure 1.

IP backhaul networks such as those of mobile networks are provisioned and managed to provide the subscribed levels of end user service. These networks are traffic engineered, and have defined mechanisms for providing differentiated services and QoS per user or flow. Policy to configure per user flow attributes in these networks have traditionally been based on monitoring and static configuration.

Currently, these networks are increasingly used for applications that demand high bandwidth. The nature of the flows and length of end user sessions can lead to significant variability in aggregate bandwidth demands and latency. In such cases, it would be useful to have a more dynamic feedback of congestion information. In addition, eNB, SGW and PGW are administrated by one mobile operator, mobile backhaul to carry IP/UDP/GTP encapsulation is regally administrated by back haul service operator. This aggregate congestion feedback could be used to determine flow handling and admission control.

```
+-----+       +-----+  Tunnel1  +-----+  Tunnel2  +-----+  Ext
|UE|-(RAN)-| eNB |----------| S-GW |----------| P-GW |------+
+-----+       +-----+  RAN   | Core  | Network  |
+-----+  Backhaul  +-----+  Network  +-----+
```

Figure 1: Example - Mobile Network and Tunnels
3.2 Network Function Virtualization Scenario

Telecoms networks contain an increasing variety of proprietary hardware appliances, leading to increasing difficulty in launching new network services, as well as the complexity of integrating and deploying these appliances in a network.

Network Functions Virtualisation (NFV) aims to address these problems by decoupling the software from dedicated hardware platforms to a range of industry standard server hardware for various network services, through IT virtualization technology that can be moved to, or instantiated in, various locations in the network as required. In this way, it is expected to provide significant benefits for network operators (reduced expenditures for network construction and maintenance) and their customers (shortened time-to-market for new network services).

Furthermore, service functions are preferred to be deployed and managed in a data center manner, rather than being inserted on the data-forwarding path between communicating peers as today. SFC WG is currently working on a new framework to cope with this highly dynamic routing problem for a network service, which requires that the relevant data traffic be traversing a group of virtualized network function nodes (VNFs), each of which could be applied at any layer within the network protocol stack (network layer, transport layer, application layer, etc.). [SFC]

As shown in Figure 2, in a SFC-enabled domain (e.g. with or across network operator’s deployed data centers), a PDP (Policy Decision Point) is the central entity which is responsible for maintaining SFC Policy Tables (rules for the boundary nodes on deciding which IP flow to traverse which service function path), and enforcing appropriate policies in SF Nodes and SFC Boundary Nodes. Beginning at the Ingress node, at each hop of a given service function path (as decided by a matched SFC policy rule/map), if the next function node is not an immediate (L3) neighbor, packet are encapsulated and forwarded to correspondent downstream function node, as shown in Figure 3.
However, using VNFs running commodity platforms can introduce additional points of failure beyond those inherent in a single specialized server, and therefore poses additional challenges on reliability. [VNFPOOL] proposes using pooling techniques in response, which requires maintaining a backup mapping among running VNF instances for a given service function, and choosing from them for a specific data flow. It is clear that it would be helpful to make more efficient use of network capacity in case of local congestion, if the choice is based on the ECN feedback as well as the running status and/or physical resources accommodation of a candidate VNF instance.
3.3 Data Center Tenancy Scenario

In the scenario of data center of multi-tenant, network resource would be shared between more than one tenants, and in order to provide functional isolation and at the same time guarantee scalability for tenants, the tunnel based isolation mechanisms, e.g. VxLAN and STT etc, are provided.

In the scenario described above, hypervisor or vSwitch would act as tunnel endpoint for the traffic between VMs, and tunnels are agnostic to VMs, in other words, the congestion indication information such as ECN flag marked by network entity of data center are agnostic to VMs. To deal with this situation, two solutions could be used:

Solution 1: Using tunnel translation, hypervisor or vSwitch marks the inner IP header according to ECN flag in outer IP header before transmits packets to VM.

Solution 2: Using the congestion control mechanism provided in this document between hypervisors or vSwitches to do congestion control for VMs’ traffic.

4. Congestion Control Model

In this section, the basic congestion control model will be provided, and each detailed aspect of this model will also be introduced in the following subsection.

The congestion control model provides network administrator with a method to manage the data traffic in its network domain. The basic model consists of the following components: Ingress, Egress, Feedback, Meter, Collector and Manager.

As shown in Figure 4, network traffic enters the tunnel through tunnel ingress, passing through en-route routers, which will mark packets according to ECN mechanism as specified in RFC3168, to tunnel egress; the egress collects the congestion level information encountered in tunnel and feeds back it to the corresponding ingress; after receiving congestion information, the ingress takes actions to control the traffic that passing through the path between the ingress and egress to reduce the congestion level in the tunnel.

At egress, a module named Meter is used to estimate the congestion level in the tunnel as described in the section above. A congestion information feedback module, called Feedback, is used to control the congestion information feedback procedure.

The metering module named Meter in the Egress node accounts the
congestion marks it receives. The Feedback module calculates the amount of congestion and feeds back the congestion information to the Ingress node. The Collector at the Ingress receives the congestion information which is fed back from the Feedback module. The Manager implements functions such as admission control and traffic engineering according to the congestion level experienced in tunnel to control the traffic to reduce the congestion level, the detailed actions taken by the Manager are out of the scope of current document.

To support traffic management and congestion information feedback in tunnel, there are mainly three issues that this document discusses: calculation of congestion level information, feeding back the congestion information from egress to ingress, and implementation of congestion control. The tunnel ingress/egress are assumed to be compliant with RFC6040 and the tunnel interior routers are compliant with RFC3168.

In addition, it should be noted that these tunnels may carry ECT or Not-ECT traffic. A well defined mechanism for aggregate congestion calculation should be able to work in the presence of all kinds of traffic and would benefit from a common feedback mechanism and protocol.

4.1 Congestion Calculation

This section discusses how to calculate congestion level experienced in the tunnel, an example of how to calculate congestion level is provided. In this document calculation of congestion in the tunnel is based on the method described in RFC 6040, Appendix C.
The egress can calculate congestion using moving averages. The proportion of packets not marked in the inner header that have a CE marking in the outer header is considered to have experienced congestion in the tunnel. Note that the packets are ECN capable and not congestion-marked before tunnel. Since routers implementing RED randomly select a percentage of packets to mark, this method can be effectively used to expose congestion in the tunnel.

When the ingress is RFC6040 compliant, the packets collected by egress can be divided into 4 categories, shown in figure 5. The tag before "|" stands for ECN field in outer header; and the tag after "|" stands for ECN field in inner header.

"Not-ECN|Not-ECN" indicates traffic that does not support ECN, for example UDP and Not-ECT marked TCP; "CE|CE" indicates ECN capable packets that have CE-mark before entering the tunnel; "CE|ECT" indicates ECN capable packets that are CE-marked in the tunnel; "ECT|ECT" indicates ECN capable packets that have not experienced congested in tunnel (or outside the tunnel).

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Not-ECN</td>
<td>Not-ECN</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>CE</td>
<td>CE</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>CE</td>
<td>ECT</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>ECT</td>
<td>ECT</td>
</tr>
</tbody>
</table>

Figure 5: ECN marking categories by outer/inner packet

Out of the total number of packets, if the quantity of CE|ECT packets is $A$, the quantity of ECT|ECT packets is $B$, then the congestion level ($C$) can be calculated as follows:

$$C = \frac{A}{A+B}$$

As an example, consider 100 packets to calculate the moving average as shown in RFC 6040, Appendix C. Say that there are 12 packets that have CE|ECT marks indicating that they have experienced congestion in the tunnel. And, there are 58 packets that have ECT|ECT marks indicating that there was no congestion in either the tunnel or elsewhere. The egress can calculate congestions as:
4.2 Data Information

This section discusses congestion-related information that should be conveyed from egress to ingress.

(1) Congestion volume. The information indicating the how much congestion has been experienced in the tunnel by traffic passing through the tunnel. Because there are both ECT packets and Not-ECT packets passing through the tunnel network, and in case of congestion, the ECT packets would be CE-marked instead of dropped and tunnel egress can be aware of these CE-marked packets; but Not-ECT packets would be dropped and tunnel egress cannot be aware of these dropped packets, so it’s hard for egress to calculate the precise number of congested packets. According to the analysis in subclause 4.1, the congestion volume is preferred in the form of percentage, e.g. 17.14%.

(2) Egress identifier. To control the traffic congestion in certain tunnel, the ingress needs to have the knowledge of which traffic should be controlled, especially for the case that the ingress establishes tunnels with different egresses. So the egress identifier should be transmitted together with congestion volume to ingress. This identifier is usually the identifier of the tunnel or the address of tunnel egress.

4.3 Congestion Feedback

This sub-section focuses on the discussion of feedback procedure. The congestion feedback procedure conveys congestion status from egress to ingress. The discussion of feedback protocol will be discussed in the next section.

To reduce the overload, caused by this procedure, on network especially in case the feedback signal goes through the same path as data traffic, the feedback will only occur when congestion happens. In other words, egress doesn’t send feedback signal if there is no congestion happens. Also egress will ignore ephemeral congestion and only feed back congestion information if the congestion level goes higher than a specified threshold (TH1) and/or lasts for a specified period of time (T1).

When egress detects congestion level higher than TH1 and for a period of T1, it sends feedback signal to ingress periodically (T2) until
the congestion level is lower than TH1.

4.4 Congestion Control

After ingress receives congestion information from egress, it will take actions to try to reduce the congestion. For example, ingress could choose to drop some packets or do certain traffic engineering etc.

Usually, network policy would have impact on what action is to be taken. For example, which packets to drop may be decided by the agreement between subscriber and network administrator. The specific choice of congestion alleviation measures taken by the ingress is out of scope of this document.

The ingress will continue to implement control actions until there is no congestion feedback from the egress.

5. Congestion Feedback Protocol

In different networks, there are always different tunnel protocols deployed. For instance, the congestion feedback can be done either by utilizing the existing tunnel protocol or using an alternative protocol. For example, in 3GPP network GTP (GPRS Tunnel Protocol)[TS29.060] is used as tunnel protocol to transmit traffic between network entities. And because GTP protocol is easy to be extended for additional information element, GTP itself would be a good choice for congestion feedback. In some other networks an independent protocol could be used for congestion feedback, for example the network using tunnel protocols such as IP-in-IP [RFC2003], GRE [RFC2784].

Currently, this section mainly focuses on the discussion of independent protocols for congestion feedback. There are two choices for such an independent protocol, one is define as a new dedicated protocol from scratch, the other one is meant to evaluate and reuse the existing protocol(s).

5.1 Properties of Candidate Protocol

To feedback congestion efficiently there are some properties that are desirable in the feedback protocol.

1. Congestion friendliness. The feeding back traffics are coexistence with other traffics, so when congestion happens in the network, the feeding back traffic should be reduced, So that feedback itself will not congest the network further when the network is
already getting congested. In other words, feedback frequency should adjust to network’s congestion level.

2. Extensibility. The authors consider that using an existing protocol, or extensions to an existing protocol is preferable. The ability of a protocol to support modular extensions to report congestion level as feedback is a key attribute of the protocol under consideration.

3. Compactness. In different situations, there may be different congestion information to be conveyed, and in order to reduce network load, the information to be conveyed should be selectable, i.e. only the required information should be possible to convey.

4. In/Out of band signal. The feedback 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.

5.2 IPFIX Extensions for Congestion Feedback

This section outlines IPFIX extensions for feedback of congestion. The authors consider that IPFIX is a suitable protocol that is reasonably easy to extend to carry tunnel congestion reporting. The Feedback module acts as IPFIX exporter, and Collector module acts as IPFIX Collector.

Since IPFIX is preferred to use SCTP as transport, it has the foundation for congestion-friendly behavior, and because SCTP allows partially reliable delivery [RFC3758] - IPFIX message channels can be tagged so that SCTP does not retransmit certain losses. This makes it safe during high levels of congestion in the reverse direction, to avoid a congestion collapse. When congestion occurs in the network, the Exporter (Egress) can reduce the IPFIX traffic. Thus the feedback itself will not congest the network further when the network is already getting congested. When the Exporter detects network congestion, it can also reduce IPFIX traffic frequency to avoid more congestion in network while being able to sufficiently convey congestion status.

Because the template mechanism in IPFIX is flexible, it allows the export of only the required information. Sending only the required information can also reduce network load.
The basic procedure for feedback using IPFIX is as follows:

1. The exporter informs the collector how to interpret the IEs in IPFIX messages using a template. The collector just accepts the template passively; which IEs to send is configured by other means that are not included in the IPFIX specification.

2. The exporter meters the traffic and sends the congestion level to the collector.

Congestion feedback using IPFIX is shown in the figures below. There are two variations to the congestion feedback model using IPFIX. In the first one shown in Figure 6(a), congestion information is sent directly from egress to ingress and ingress makes decisions according to this information. In the second case shown in Figure 6(b), congestion information is sent to a mediation controller instead of tunneling directly to ingress; the controller is in charge of making decisions according to network congestion and controls the behavior of ingress, for example, reducing traffic or forbidding new traffic flows. In this model, the congestion information from egress to controller is conveyed by IPFIX, but how the controller controls the behavior of ingress is out of scope of this document.

(a) Direct Feedback.
To support feeding back congestion information, some extensions to the IPFIX protocol are necessary. According to the definition of congestion-related information defined in "Data Mode" section, new IEs conveying congestion level is defined for IPFIX.

**Definition of new IE indicating congestion level.**

**Description:**
- The congestion level calculated by exporter.
- Abstract Data Type: float32
- Data Type Semantics: quantity
- ElementId: TBD.
- Status: current

The example below shows how IPFIX can be used for congestion feedback.

**(1) Sending Template Set**
The exporter use Template Set to inform the collector how to interpret the IEs in the following Data Set.

```plaintext
+------------------------+--------------------+
| Set ID=2               | Length=n            |
+------------------------+--------------------+
| Template ID=257        | Field Count=m      |
+------------------------+--------------------+
| exporterIPv4Address=130| Field Length=4     |
+------------------------+--------------------+
| collectorIPv4Address=211| Field Length=4      |
+------------------------+--------------------+
| CongestionLevel=TBD1   | Field Length=2     |
+------------------------+--------------------+
| Enterprise Number=TBD2 |
+------------------------+--------------------+
```
(2) Sending Data Set The exporter meters the traffic and sends the congestion information to collector by Data Set.

```
<table>
<thead>
<tr>
<th>Set ID=257</th>
<th>Length=n</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>192.0.2.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>192.0.2.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1714</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 7: IPFIX Congestion Flow

Before sending congestion information to collector, the exporter sends a Template set to Collector. The Template set specifies the structure and semantics of the subsequent Data Set containing congestion-related information. The Collector understands the Data Sets that follow according to Template Set that was sent previously. The exporting Process transmits the Template Set in advance of any Data Sets that use that Template ID, to help ensure that the Collector has the Template Record before receiving the first Data Record. Data Records that correspond to a Template Record may appear in the same and/or subsequent IPFIX Message(s).
The Exporter meters the traffic passing through it and generates flow records. At this point, the Exporter may cache the records and then send congestion cumulative information to the collector. When Exporter detects that the network is heavily congested, it can change the feedback frequency to avoid adding more congestion to network.

When receiving congestion related information, the Collector will make decisions to control the traffic entering the tunnel to reduce tunnel congestion.

5.3 Other Protocols

A thorough evaluation of other protocols have not been performed at this time.

6. Benefits

This section provides a short discussion about what benefits the tunnel congestion control would bring.

Tunnel congestion control is a kind of local congestion control, where each tunnel is treated as an independent administrative domain in terms of congestion feedback and control, and it only responds to the congestion happened in the tunnel. The tunnel congestion control is complementary with e2e ECN control.

The tunnel congestion feedback provides the network administrator with network congestion level information that can be used as an input for it local network management rather than relying solely on the e2e congestion control or blind traffic throttling. If the tunnel is congested it will be a waste of resource to allow new traffic to enter, because they may eventually get dropped in the tunnel. It’s more efficient to have a control on new traffic at ingress.

7. Security Considerations

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

IANA assignment of parameters for IPFIX extension may need to be considered in this document.
9. References

9.1 Normative References


9.2 Informative References

[TS29.060]3GPP TS 29.060: "General Packet Radio Service (GPRS); GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface".

Wei                      Expires April 11, 2015                [Page 19]
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