

Transport Area Working Group
Internet-Draft
Updates: 3819 (if approved)
Intended status: BCP
Expires: August 28, 2013

B. Briscoe
BT
J. Kaippallimalil
Huawei
P. Thaler
Broadcom Corporation
February 24, 2013

Guidelines for Adding Congestion Notification to Protocols that
Encapsulate IP
draft-briscoe-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 August 28, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

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

Table of Contents

1. Introduction	3
1.1. Scope	5
2. Terminology	5
3. Modes of Operation	7
3.1. Feed-Forward-and-Up Mode	7
3.2. Feed-Up-and-Forward Mode	9
3.3. Feed-Backward Mode	10
3.4. Null Mode	12
4. Feed-Forward-and-Up Mode: Guidelines for Adding Congestion Notification	12
4.1. IP-in-IP Tunnels with Tightly Coupled Shim Headers	13
4.2. Wire Protocol Design: Indication of ECN Support	13
4.3. Encapsulation Guidelines	15
4.4. Decapsulation Guidelines	16
4.5. Sequences of Similar Tunnels or Subnets	18
4.6. Reframing and Congestion Markings	18
5. Feed-Up-and-Forward Mode: Guidelines for Adding Congestion Notification	19
6. Feed-Backward Mode: Guidelines for Adding Congestion Notification	20
7. IANA Considerations (to be removed by RFC Editor)	21
8. Security Considerations	21
9. Conclusions	21
10. Acknowledgements	21
11. Comments Solicited	22
12. References	22
12.1. Normative References	22
12.2. Informative References	22
Appendix A. Outstanding Document Issues	25
Appendix B. Changes in This Version (to be removed by RFC Editor)	25

1. Introduction

Explicit Congestion Notification (ECN [RFC3168]) is defined in the IP header (v4 & v6) 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.

ECN removes nearly all congestion loss and it cuts delays for two main reasons: i) it avoids the delay when recovering from congestion losses, which particularly benefits small flows, making their completion time predictably short [RFC2884]; and ii) 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). The latter delay 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 large subnetworks with IP-aware nodes only at the edges. These networks are often designed so that it is rare for interior queues to overflow. However, this has often only been possible because the original design of TCP did not scale, and fixes (e.g. [RFC1323]) proved hard to deploy. 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. However, the above benefits of ECN can only be fully realised if support for ECN is added to the relevant subnetwork technology, as well as IP. When a lower layer queue drops a packet 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.

Propagation of ECN is defined for MPLS [RFC5129], and is being defined for TRILL [trill-rbridge-options], 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].

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.

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 3, 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 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.

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.

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. 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).

3.1. Feed-Forward-and-Up Mode

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

In these cases, ECN may best be supported by standardising explicit notification of congestion into the lower layer protocol that carries the data forwards. 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).

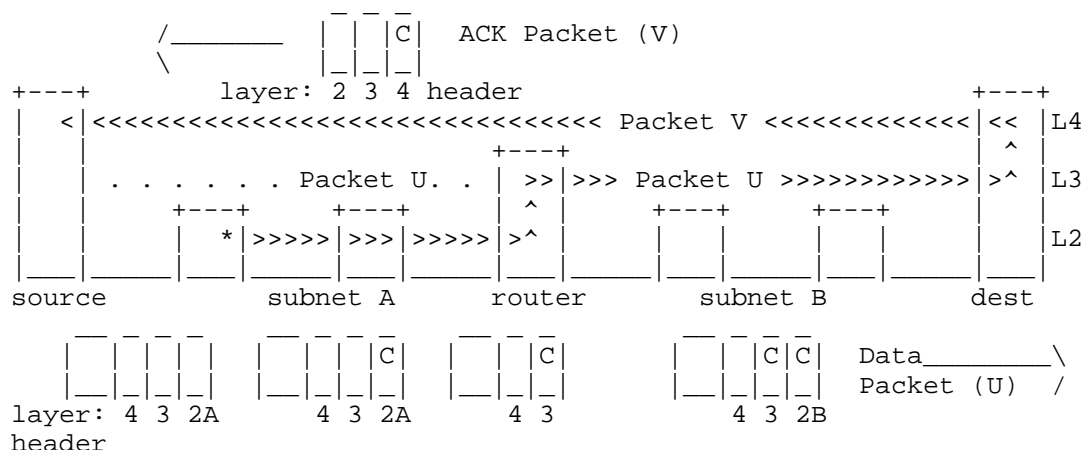


Figure 1: Feed-Forward-and-Up Mode

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].

3.2. Feed-Up-and-Forward Mode

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

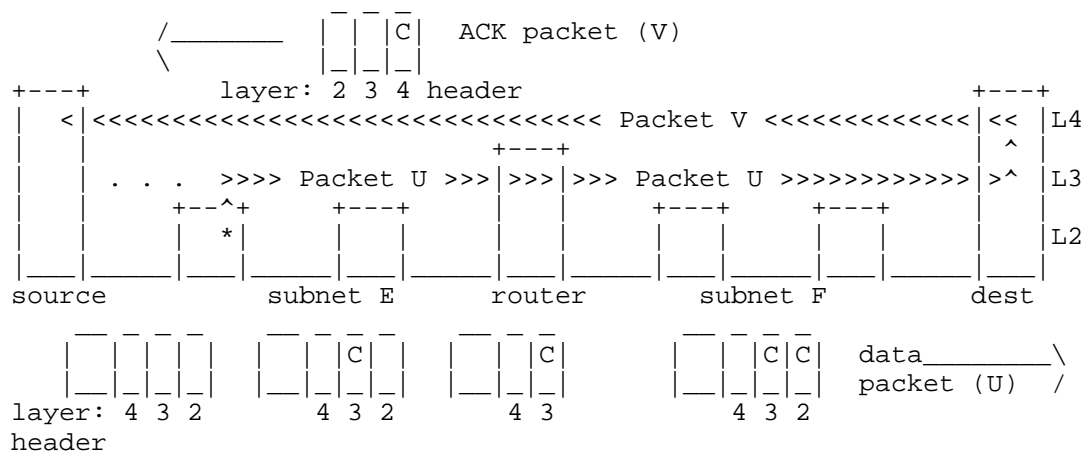


Figure 2: Feed-Up-and-Forward Mode

By comparing Figure 2 with Figure 1, it can be seen that subnet E (perhaps a subnet of layer-3 Ethernet switches) works in feed-up-and-

forward mode by notifying congestion directly into L3 at the point of congestion, even though the congested switch does not otherwise act at L3. In this example, the technology in subnet F (e.g. MPLS) does support ECN natively, so when the router adds the layer-2 header it copies the ECN marking from L3 to L2 as well.

3.3. Feed-Backward Mode

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

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

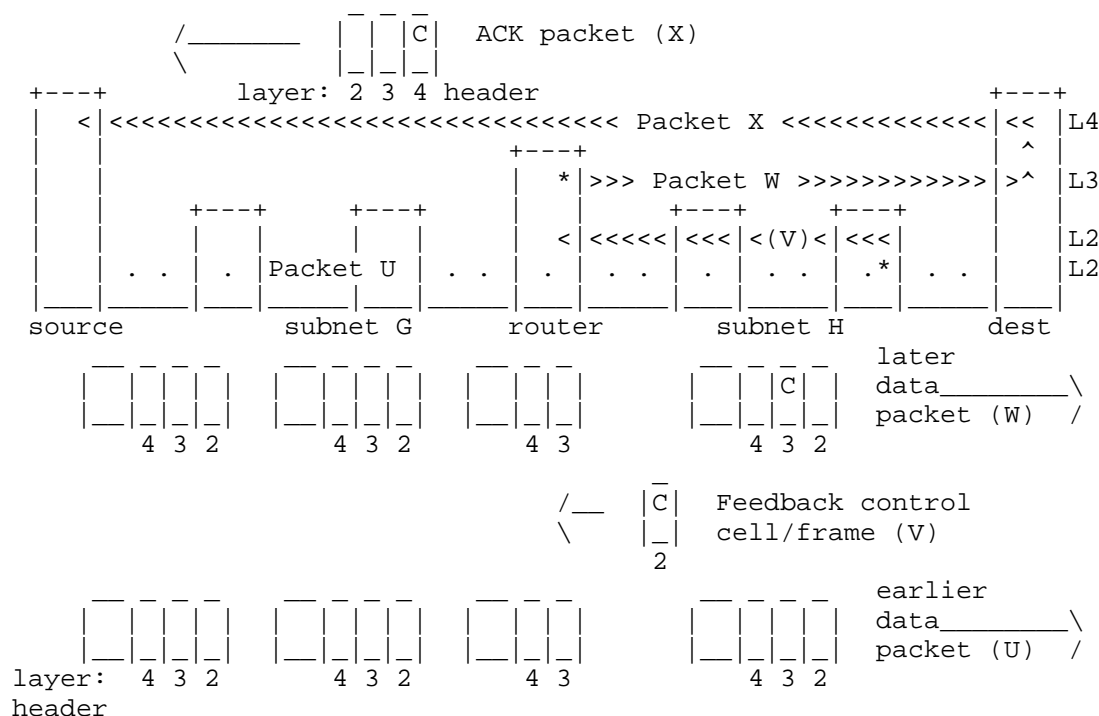


Figure 3: Feed-Backward Mode

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

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

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

3.4. Null Mode

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

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

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

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

The rest of this section is structured as follows:

- o Section 4.1 addresses the most straightforward cases, where [RFC6040] can be applied directly to add ECN to tunnels that are effectively the same as IP-in-IP tunnels.
- o The subsequent sections give guidelines for adding ECN to a subnet technology that uses feed-forward-and-up mode like IP, but it is not so similar to IP that [RFC6040] rules can be applied directly. Specifically:
 - * Sections 4.2, 4.3 and 4.4 respectively address how to add ECN support to the wire protocol and to the encapsulators and decapsulators at the ingress and egress of the subnet.
 - * Section 4.5 deals with the special, but common, case of sequences of tunnels or subnets that all use the same

technology

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

4.1. IP-in-IP Tunnels with 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:

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

4.2. Wire Protocol Design: Indication of ECN Support

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 4.4).

It was only appropriate to define such an incremental deployment strategy because MPLS is targeted solely at professional operators, who can be expected to ensure that a whole subnetwork is consistently configured. This strategy might not be appropriate for other link

technologies targeted at zero-configuration deployment or deployment by the general public (e.g. Ethernet). For such 'plug-and-play' environments it will be necessary to invent a failsafe approach that ensures congestion markings will never fall into black holes, no matter how inconsistently a system is put together. Alternatively, congestion notification relying on correct system configuration could be confined to flavours of Ethernet intended only for professional network operators, such as 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.

4.3. Encapsulation Guidelines

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 [trill-rbridge-options])
 - * 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]).

4.4. Decapsulation Guidelines

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 4.3 had been followed, it should not be possible to have a less severe indication of congestion in the outer than in the inner. It MAY be appropriate to log unexpected combinations of headers and possibly raise an alarm.

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

4.5. Sequences of Similar Tunnels or Subnets

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

In such cases, it would be sufficient for every encapsulation and decapsulation in the chain to comply with RFC6040. Alternatively, as an optimisation, a node that decapsulates a packet and immediately re-encapsulates it for the next tunnel MAY copy the incoming outer ECN field directly to the outgoing outer and the incoming inner ECN field directly to the outgoing inner. Then the overall behavior across the sequence of tunnel segments would still be consistent with RFC 6040.

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

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

4.6. Reframing and Congestion Markings

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.

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

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

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

Nonetheless, configuring a layer-3 switch to look for an ECN field in an encapsulated IP header is a useful optimisation. If the implementation follows the guidelines below, this optimisation does not have to be confined to a controlled environment such as within a data centre; it could usefully be applied on any network--even if the operator is not sure whether the above issues will never apply:

1. If a native lower-layer congestion notification mechanism exists for a subnet technology, it is safe to mix feed-up-and-forward with feed-forward-and-up on other switches in the same subnet. However, it will generally be more efficient to use the native mechanism.
2. The depth of the search for an IP header SHOULD be limited. If an IP header is not found soon enough, or an unrecognised or unreadable header is encountered, the switch SHOULD resort to an alternative means of signalling congestion (e.g. drop, or the native lower layer mechanism if available).

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

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

It can be seen from Section 3.3 that congestion notification in a subnet using feed-backward mode has generally not been designed to be directly coupled with IP layer congestion notification. The subnet attempts to 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 does at least work beneath IP, but it 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.

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

This memo includes no request to IANA.

8. Security Considerations

{ToDo}'

9. Conclusions

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

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

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

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

10. Acknowledgements

Thanks to Gorrry Fairhurst for extensive initial reviews. Michael Welzl 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 author.

11. 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.

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.
- [RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.
- [RFC4774] Floyd, S., "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field", BCP 124, RFC 4774, November 2006.

12.2. Informative References

- [ATM-TM-ABR] Cisco, "Understanding the Available Bit Rate (ABR) Service Category for ATM VCs", Design Technote 10415, June 2005.
- [Buck00] Buckwalter, J., "Frame Relay: Technology and Practice", Pub. Addison Wesley ISBN-13: 978-0201485240, 2000.
- [DCTCP] Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and M. Sridharan, "Data Center TCP (DCTCP)", ACM SIGCOMM CCR 40(4)63--74, October 2010, <<http://portal.acm.org/citation.cfm?id=1851192>>.
- [GTPv1] 3GPP, "GPRS Tunnelling Protocol (GTP) across

- the Gn and Gp interface", Technical Specification TS 29.060.
- [GTPv1-U] 3GPP, "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", Technical Specification TS 29.281.
- [GTPv2-C] 3GPP, "Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C)", Technical Specification TS 29.274.
- [IEEE802.1Qah] IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Virtual Bridged Local Area Networks--Amendment 6: Provider Backbone Bridges", IEEE Std 802.1Qah-2008, August 2008, <<http://www.ieee802.org/1/pages/802.1ah.html>>.
- (Access Controlled link within page)
- [IEEE802.1Qau] Finn, N., Ed., "IEEE Standard for Local and Metropolitan Area Networks--Virtual Bridged Local Area Networks - Amendment 13: Congestion Notification", IEEE Std 802.1Qau-2010, March 2010, <<http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5454061>>.
- (Access Controlled link within page)
- [ITU-T.I.371] ITU-T, "Traffic Control and Congestion Control in B-ISDN", ITU-T Rec. I.371 (03/04), March 2004.
- [RFC1323] Jacobson, V., Braden, B., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.
- [RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, October 1994.
- [RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.
- [RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point

- Tunneling Protocol", RFC 2637, July 1999.
- [RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, August 1999.
- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.
- [RFC2884] Hadi Salim, J. and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", RFC 2884, July 2000.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, January 2008.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.
- [RFC6633] Gont, F., "Deprecation of ICMP Source Quench Messages", RFC 6633, May 2012.
- [RFC6660] Briscoe, B., Moncaster, T., and M. Menth, "Encoding Three Pre-Congestion Notification (PCN) States in the IP Header Using a Single Diffserv Codepoint (DSCP)", RFC 6660, July 2012.
- [trill-rbridge-options] Eastlake, D., Ghanwani, A., Manral, V., and C. Bestler, "RBridges: Further TRILL Header Extensions", draft-ietf-trill-rbridge-options-07 (work in progress), June 2012.
- [vxlan] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks",

draft-mahalingam-dutt-dcops-vxlan-03 (work
in progress), February 2013.

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. [GF] Impact of Diffserv on alternate marking schemes (referring to RFC3168, RFC4774 & RFC2983)
3. Consider whether an IETF Standard Track doc will be needed to Update the IP-in-IP protocols listed in Section 4.1--at least those that the IETF controls--and which Area it should sit under.
4. Guidelines referring to subnet technologies should also refer to tunnels and vice versa.
5. Check that guidelines allow for multicast as well as unicast.
6. Security Considerations

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

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://bobbbriscoe.net/>

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

EMail: john.kaippallimalil@huawei.com

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

EMail: pthaler@broadcom.com

TSVWG
Internet-Draft
Intended Status: Informational
Expires: August 25, 2013

K. Carlberg
G11
P. O'Hanlon
UCL
Feb 25, 2013

Reactions to Signaling from ECN Support for RTP/RTCP
<draft-carlberg-tsvwg-ecn-reactions-04.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). 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 25, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Abstract

This document presents an examination of various responses to Congestion Experience (CE) notifications by real time applications that have negotiated end-to-end support of Explicit Congestion Notification (ECN). This document is a follow-on effort of [rfc6679], which specifies the signaling used to provide ECN support for RTP/RTCP flows.

1. Introduction

This document presents an examination of various responses to Congestion Experience (CE) notifications by real time applications that have negotiated end-to-end support of Explicit Congestion Notification (ECN). [rfc6679] defines the signaling for support of ECN by RTP based sessions and also covers the case where a set of nodes do not respond to CE notifications. A more detailed discussion about how back-off algorithms can be achieved, as well as other potential reactions, is viewed as out of scope of that document and may be addressed by a companion document.

1.1 Background

ECN is a mechanism used to explicitly signal the presence of congestion without relying on packet loss. It was initially designed using a dual layer signaling model; negotiation and feedback at the transport layer, and downstream notification of congestion at the network layer. For IP, a new two bit field was used to both indicate the successful negotiated support for ECN signaling, as well as indicate the presence of congestion via the CE flag. In the case of TCP [rfc3168], a new TCP header flag was defined that provides upstream end-to-end indication of congestion occurring somewhere along the downstream path.

There should be no difference in congestion response if ECN-CE marks or packet drops are detected. However it is noted that there MAY be other reactions to ECN-CE specified in the future. Such an alternative reaction MUST be specified and considered to be safe for deployment under any restrictions specified. We specify such an alternative in this document.

With respect to ECN for TCP, [rfc3168] specifies an indication of congestion, but it does so once per Round Trip Time (RTT). [rfc6679] is an effort that proposes a finer grained notification reflecting a more accurate indication of the number of ECN marked packets received within one RTT. It should be noted that there is also other on going work to provide more accurate ECN feedback information for TCP [draft-tcpm-accecn-reqs].

1.2 Terminology and Abbreviations

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 [RFC2119].

2. Issues

The initial discussions and presentation of [draft-rtp-ecn] produced a consensus that the specification of signaling was to be done within the AVTcore working group, and any subsequent discussion on end-to-end reactions to the signaling would be accomplished in the Transport Services (TSV) working group. This draft satisfies the latter effort.

Another issue that needs to be recognized is that the reactions to CE in the context of [rfc6679] are the responsibility of the application. This is in contrast to ECN support for TCP, where explicit signaled feedback of, and reaction to, CE is kept transparent to the application. The issue of placing the feedback responsibility in the application is that each application needs to add specific support for that reaction. On the other hand, multiple reactions may be considered by the application. For this reason, [rfc6679] states the need for a default congestion control reaction that MUST be supported. Section 3 through 5 expands on this topic.

3. Congestion Control Algorithms

The transport of any data flow across the Internet produces a need for some form of congestion control to attain a suitable share of the capacity of the path through a network. Most of the existing work on realtime congestion control algorithms has been rooted in TCP-friendly approaches but with smoother adaptation cycles. TCP congestion control is unsuitable for interactive media for a number of reasons including the fact that it is loss-based so it maximizes the latency on a path, it changes its transmit rate to quickly for multimedia, and favors reliability over timeliness. In the case of real time media transport, one requires:

Smoother rate variation: (than for bulk data) to accommodate the underlying media flow's characteristics.

Low latency: Maintaining latencies sufficient to be usable, where 150ms is understood to be a good target [ITU.G114.2003].

Burst handling: Ability to handle bursts due to the nature of the media and codec (e.g. I-frames etc)

3.1 TCP Friendly Rate Control (TFRC)

TFRC has a smoother response to congestion than TCP-like approaches, thus making it more suitable for real-time interactive multimedia applications. It has been cited in a number of other documents within the IETF for use with UDP and media flows [rfc3714, bcp145] and is seeing full and partial deployment in related solutions such as Empathy/Farsight, and GoogleTalk [googl].

However it should be noted that TFRC is only recommended for real-time media use with ECN response. TFRC is not recommended for non-ECN paths due to its loss based operation which leads to full queues with maximised latencies. It is assumed that ECN markings will usually occur with lower queue occupancy and thus lower latency. However it is understood that ECN marks may not provide for sufficiently low latencies in some situations so other congestion control solutions would be preferable.

[rfc4342] specifies the profile for TFRC for use in the Datagram Congestion Control Protocol (DCCP) [rfc4340] for a half connection. A DCCP half connection is defined as application data sent downstream with corresponding acknowledgements sent upstream. These half-connections can be realized in the form of one-way pre-recorded media, one-way live media, or two-way interactive. A perceived drawback in this profile concerns its application to interactive media that use small packets. [RFC4828] is an experimental protocol defining a variation of TFRC used to address this drawback and achieve the same bandwidth as a TCP flow using packets of size 1500 bytes.

[rfc6679] is an standard that specifies how RTP flows can be supported using the RTP/AVPF profile and the general RTP header extension mechanism.

3.2 Related Work

3.2.1 3GPP

Outside of this previous and on-going work with TFRC, it is understood that some parties have issues with the behavior of TFRC under certain conditions. A notable mention of this is made in the 3GPP's document on IP Multimedia Subsystem (IMS) Media handling and interaction [TR26.114], where it is mentioned:

"Note that for IMS networks, which normally have nonzero packet loss and fairly long round-trip delay, the amount of bitrate reduction specified in RFC 3448 is generally too restrictive for video and may, if used as specified, result in very low video bitrates already at (for IMS) moderate packet loss rates."

Though it is unclear exactly what the 3GPP community consider as too restrictive and whether some alteration of the response may be suitable. It should be noted that the 3GPP document only referred to an older version of TFRC defined in [RFC3448]. Given that the current version of TFRC [RFC5348] has made significant changes to the idle and data-limited responses it is unclear whether their assessment is relevant to current TFRC implementations.

Furthermore the specification [TR26.114] only outlines a rudimentary approach to congestion control, providing an example of a 60% back-off reaction to loss within an RTCP reporting period. The proposed signalling employs Temporary Maximum Media Stream Bit Rate Request (TMMBR) [RFC5104] and Codec Mode Request (CMR) [RFC4867] for video and audio respectively, which would only provide for very basic rate control if used as specified. We note that [TR26.114] specifies terminal behavior, while [TS36.300] specifies base station behaviour, though neither specify any standardised congestion control approach.

It is understood that there are a number of proprietary and patented approaches that provide more sophisticated response in the case of 3G/LTE, but since these are neither endorsed nor standardized this document advocates a standardized approach such as TFRC.

We also acknowledge that there are many congestion control algorithms available for implementers to choose from, with a subset that are specifically suited to real time media transmission. However, given a variety of real time applications and their various characteristics (sender-only broadcast, interactive unicast, etc), we need to expand the notion of how back-off can be achieved. Hence, the focus needs to be on an output that would resemble the characteristics of TFRC.

3.2.2 RTCweb

Within the RTCweb Working Group the need for a more media friendly congestion control mechanism has been made apparent. Currently, TFRC is perceived as having deficiencies (e.g. its loss-based design, lack of cross-stream congestion control functionality etc) that make it an incomplete or insufficient solution for the envisioned RTCWEB media flows. The RTP Media Congestion Avoidance Techniques (rmcat) working group has now been formed which aims to lead to the formation of a working group on these issues. The group aims to develop one or more congestion control algorithms, associated extensions, and evaluation criteria. Furthermore it has been proposed that certain practices, such as 'circuit-breaker' conditions, to provide operational limits on congestion control algorithms, and feedback messages, may be tackled in other groups such as AVTCORE and AVTEXT respectively.

Thus there is some movement to attempt to develop new algorithms better suited to media transport, but these efforts will clearly take a considerable time to reach fruition.

3.3 ECN response

As mentioned above and in accordance to [rfc3168], the actual response to the reception of an ECN-CE marked packet MUST normally be the same as that of a lost packet. However there are a number of contexts where one

may also be interested in more varied approaches. We expand on this in Section 5 below.

4. Application Layer Congestion Response

Whilst the congestion control algorithm may decide to alter the rate at which the application should operate, in the case of media applications this process is not as straightforward as the case of bulk data. The different media engines and codecs in use may only have limited adaptation ranges, thus, this limitation needs to be a consideration when adapting the rate. Furthermore the application needs to be aware of the capability of the specific codecs in terms of their ability to switch configuration mid-stream (without loss of fidelity), which may impose further limits on the modes of operation.

One approach for achieving a lower generation of data is through reduced sampling of the media (e.g., voice or video). In the case of video, this may also involve slower frame rates. Specific recommendations that describe how applications should respond to congestion in the context of supporting the algorithmic characteristics of a congestion control algorithm are outside the scope of this document.

5. Other Reactions

In addition to the activation of congestion control algorithm, other reactions can be used or leveraged by an application in response to CE. We divide these other potential reactions into three categories: signaling, fault tolerance, and reduction. In the first two cases, we note that these other reactions are considered symmetric because they require downstream peer support. We also point out that activation of other reactions represents an example of an on-demand and as-needed approach in responding to CE.

In each case, we discuss issues that should be considered when contemplating a different reaction in the presence of CE feedback.

5.1 Signaling

5.1.1 RSVP

The resource Reservation Protocol (RSVP) can be used to signal a desired set of path characteristics (e.g., bandwidth, delay) in response to CE feedback [rfc2205]. Its operation is based on the use of PATH messages sent downstream hop-by-hop from the source to a destination that specify requested forwarding characteristics. In return, the destination sends a hop-by-hop RESV message upstream towards the source confirming the resources that have been reserved for that flow.

[rfc3181] defines a priority policy element that specifies both an allocation and defending priority. This dual specification supports the use of preemption of existing reservations. [draft-priority-rsvp] is a work-in-progress that defines a new policy element that only conveys priority during reservation establishment. This latter effort also presents several reservation models, including one that describes engineered resources set aside for priority users.

5.1.1.1 Issues

As discussed in [rfc-3583], RSVP presents a difficult challenge of establishing state and effectively and efficiently migrating it during roaming in mobile environments. Its soft state design allows the protocol to attempt re-establishment of reserved resources along new path(s), but there is no guarantee that resources along the new path will be available. In addition, there is at least 1 RTT of delay and the delta in initiating a new PATH message that delays reservation establishment.

Some user groups, such as those found in the military, make a distinction between mobile and transportable environments. The former case resembles scenarios attributed to Mobile IP. The latter case is characterized by wireless hosts operating in a new location, but never moving to the extent that new paths through a network need to be established. In this latter example, the challenges of RSVP in a wireless environment are diminished. In addition, these environments tend to involve a single administrative control of both hosts and routing/forwarding nodes within a network infrastructure.

RSVP is associated with a means of retaining a minimal bound of forwarding characteristics per flow, or aggregate of flows. As such, it can be considered to run contrary to the objectives of ECN. However, in cases where some flows must be reserved, CE feedback could be used to signal the need to lower a pre-existing killer app reservation.

5.1.2 Differentiated Services

Unlike RSVP and its use of a separate signaling mechanism to reserve resources, Differentiated Services (diff-serv) uses code points within the IP header to convey the forwarding behavior of that packet [rfc2474]. This may range from various drop precedence values to a code point that signifies low delay and low loss (i.e., characteristics attributed to real time flows).

As in the case of RSVP, applications could rely on the reception of CE feedback to initiate a subsequent setting of diff-serv code points to provide additional protection or explicit association of forwarding characteristics of a given flow of packets. In addition, the setting of

diff-serv code points would be done on an as-needed basis in reaction to CE feedback. Recommendations concerning specific diff-serv values are outside the scope of this document.

5.1.2.1 Issues

Given the ease by which applications or middle boxes can set diff-serv code points, the issue of trusting values other than best effort can become problematic when hosts and routing/forwarding nodes are not associated with a single administrative authority.

As in the case of RSVP, the effectiveness of diff-serv is dependent on the number of nodes along a path that support the protocol. Thus, as opposed to a single end-point reaction to CE feedback, differentiated services requires additional support in the network to either increase or decrease the probability of traffic being forwarded to its destination.

A symbiotic capability to consider is the use of on-demand/as-needed diff-serv code points to trigger downstream actions by the network. A specific example would be a diff-serv code point sent in reaction to CE feedback that could trigger alternate path routing via MPLS.

5.2 Fault Tolerance

Fault tolerance is another category of reactions that may be used by applications in response to CE feedback. In some cases, these efforts may contribute to an increase in traffic load in order to add protection and resiliency to a flow.

Redundant Transmissions: This approach is based on a source sending duplicate payloads that can be used to compensate for lost packets. Its positive value may emerge in cases where a path has several downstream congestion points that increase the probability that a packet will be dropped instead of marked as CE and forwarded downstream.

Application Layer Forward Error Correction (FEC): This approach also adds additional overhead to the flow in order to compensate for potential packet loss. And as the case of redundant transmissions, the value of this approach can be realized when there exists multiple downstream congestion points that increase the probability of dropping packets. However, the impact of the overhead is minimized by having one (or a few) additional packet(s) used to compensate for the loss of a set of packets.

Codec Swapping: This approach involves changing codecs to either reduce load or achieve an improvement in compensating for lost packets. Depending on the codec, the reduction of load may be a simple step

function, or it may involve a gradual and variable reduction in load based on the rate of congestion feedback received by the source.

Interweaving packets: To Be Done (based on research at UCL)

5.2.1 Issues

The use of redundant transmissions or FEC produces a detrimental impact of contributing to an increase in load and the measure of congestion that triggers CE feedback. In the case of FEC, additional delay is typically incurred through the generation of X amount of erasure packets for each set of original source packets. And while an initial increase in QoS may be observed for these flows, the overall rate of congestion can be expected to increase.

Swapping codecs based on the reception of CE feedback has the positive affect of reducing load at the risk of reducing perceived QoS by the user. As in the case of all options described above regarding fault tolerance, the ability to change to a different codec is depending on end-to-end peer support. In addition, there is no assurance that the different codec reduces load in relation to the amount of congestion experienced over time.

5.3 Alternative Reaction for Emergency Communications

As mentioned in [rfc6679], the default reaction on the reception of these ECN-CE marked packets MUST be to provide the congestion control algorithm with a congestion notification that triggers the algorithm to react as if packet loss had occurred. There MAY be an alternative reaction if it is considered safe for deployment. An example of the need for an alternative reaction would be the case of Emergency Telecommunications Service (ETS) [rfc3689, rfc4190], where an improvement in QoS or a higher probability of session establishment and forwarding of traffic is of high interest.

It is proposed that certain authorized ETS flows may be permitted to employ either a substantially less aggressive back-off algorithm than the default algorithm, or some level of exemption from reacting to ECN marked packets. This alternative reaction will benefit these flows as the marks would normally be considered as equivalent to lost packets, which would effectively increase the loss level, which in turn will generally result in the reduction of flow rate. This applies to all flows that utilize some form of the rate control that is inversely proportional to the loss rate, which includes TCP-like algorithms or equation-based approaches.

Simulations of the use of ECN exemption with TFRC and have found that it has limited effect on the normal flows with low numbers of exempt flows. A half-dumbbell network was used with a RED router queue configured using the

settings recommended by Sally Floyd. The candidate flows are 1Mbit/s each with a backhaul 100Mbit/s link. In the standard case where 1% of flows would be exempt the remaining flows achieve 99.99% of the bandwidth that they would achieve without the presence of the exempt flows. This is what would be expected from the simple calculation of the allocation, given that the exempt flows achieve their full rate (1Mbit/s); With 100 normal plus 1 exempt flow, assuming that the except flow uses 1Mbit/s, the remaining capacity is 99Mbit/s which is divided between the 100 normal flows. Whilst when 101 normal flows are run over the 100Mbit/s link they would have to share it evenly, so it work

s out thus: $((99/100)/(100/101))*100=99.99\%$. In the case of 5% exempt flows then the proportion is very slightly lower at $((95/100)/(100/105))*100=99.75\%$. Bot

h these calculations are borne out in the simulation runs.

The level of exemption employed can be altered in a number of ways. Two simple approaches would be to either set a threshold number of ECN marked packets tha

t could be considered as a loss, and another approach would be to set a percentage threshold of ECN marked packet that would be considered as a loss.

It should be noted that in the simulations the end-to-end delay of the packets within the flows was monitored and the relative delay of the exempt flows apparently rises somewhat when exemption is enacted. However what is actually occurring is that the 'normal' flows are reducing their throughput and are thu

s reducing their latency somewhat. There is normally some limited latency when using loss-based techniques such as TFRC because it fills the queues to ascertain the link capacity and maintains that level of delay throughout a session. However the level of latency is clearly limited by the queue sizes in the network and on media specific links these queue sizes are typically quite small, so the resulting latency is limited.

Furthermore in the case where media flows employing TFRC, or any other congestion control algorithm (e.g. delay-based), are sharing a bottleneck link with TCP flows then the queues will be filled by the TCP flows and the latency will be kept near or at a their maximum despite any other flows.

5.3.1 Issues

To Be Done

6. IANA Considerations

This document requires no actions from IANA.

7. Security Considerations

The reliance on accurate and un-modified RTCP information means that SRTP needs to be used, or any other mechanism that helps prevent modification of RTCP feedback packets.

8. Acknowledgements

TBD

9. References

9.1 Normative

- [rfc2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [rfc2205] Braden, B., et. al., "Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification", RFC 2205, September 1997
- [rfc2209] Braden, R., L. Zhang, "Resource Reservation Protocol (RSVP) Version 1 Message Processing Rules", RFC2209 September 1997
- [rfc2474] Nichols, K., et. al., "Definition of the Differentiated Services Field in the IPv4 and IPv6 Headers", RFC 2474, December 1998
- [rfc3168] Ramakrishnan, K., et. al., "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September, 2001
- [rfc3181] Herzog, S., "Signaled Preemption Priority Policy Element", RFC 3181, October 2001
- [rfc3448] Handley, M., et. al., "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003
- [rfc3583] Chaskar, H., "Requirements of a Quality of Service (QoS) Solution for Mobile IP", RFC 3583, September 2003
- [rfc4867] Sjöberg, J., et. al., "RTP Payload Format and File Storage Format for the AMR and AMR-WB Audio Codecs", RFC 4867, April 2007
- [rfc5104] Wenger, S., et. al., "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008
- [rfc6679] Westerlund, M., et. al., "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, IETF, Aug 2012

9.2 Informative

- [draft-rtp-tfrc] Gharai, L., C. Perkins, "RTP with TCP Friendly Rate Control", work-in-progress, Sept 2011
- [draft-tcpm-accecn-reqs] M. Kuehlewind, R. Scheffenegger, "Problem Statement and Requirements for a More Accurate ECN Feedback", work-in-progress, Feb 2013
- [Googl] http://code.google.com/apis/talk/call_signaling.html
- [tr26.114] "IMS; Multimedia telephony; Media Handling and Interaction", 3GPP, version 10, April 2011
- [ts36.300] "E-UTRA and E-UTRAN Overall Description, Stage 2", 3GPP, Release 10, September, 2011
- [rfc4340] Kohler, E., et. al, Datagram Congestion Control Protocol (DCCP), RFC4340, March 2006
- [rfc4342] Floyd, S., et. al., "Profile for DCCP Congestion Control ID 3: TFRC", RFC 4342, March 2006
- [rfc4828] Floyd, S., E. Kohler, "TFRC: The Small Packet Variant", RFC 4828, April 2007
- [rfc3689] Carlberg, K., Atkinson, R., "General Requirements for Emergency Telecommunications Service (ETS)", RFC 3689, February 2004
- [rfc4190] Carlberg, K. et, al., "Framework for Supporting Emergency Telecommunications Service (ETS) in IP Telephony", RFC 4190, November 2005
- [rfc3714] Floyd, S., Kempf, J., "IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet", RFC 3714, March 2004
- [bcp145] Eggert, L., Fairhurst, G., "Unicast UDP Usage Guidelines for Application Designers", RFC 5405, BCP 145, November 2008
- [ITU.G114.2003]
International Telecommunications Union, "One-way transmission time", ITU-T Recommendation G.707, May 2003.

Author's Addresses

Piers O'Hanlon

University of Oxford
Oxford Internet Institute
1 St Giles
Oxford OX1 3JS
United Kingdom

Email: piers.ohanlon@oii.ox.ac.uk

Ken Carlberg
G11
1600 Clarendon Blvd
Arlington VA
USA

Email: carlberg@g11.org.uk

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 31, 2013

S. Dhesikan
Cisco
D. Druta, Ed.
ATT
P. Jones
J. Polk
Cisco
November 27, 2012

DSCP and other packet markings for RTCWeb QoS
draft-dhesikan-tsvwg-rtcweb-qos-00

Abstract

Many networks, such as service provider and enterprise networks, can provide per packet treatments 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 31, 2013.

Copyright Notice

Copyright (c) 2012 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

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, when attempting to avoid congestion by marking certain traffic flows, say all audio or all audio and video, marking too many audio and/or video flows for a given network's capacity 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 draft proposes how a browser and other VoIP applications can mark packets. This draft does not contradict or redefine any advice from previous IETF RFCs but simply provides a simple set of recommendations for implementors based on the previous RFCs.

There are some environments where priority markings frequently help. These include:

1. If the congested link is the broadband uplink in a Cable or DSL scenario, often residential routers/NAT support preferential treatment based on DSCP.
2. If the congested link is a local WiFi network, marking may help.
3. In some cellular style deployments, markings may help in cases where the network does not remove them.

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. 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. Future drafts may define mechanisms for site specific mappings to override the values provided in this draft.

This draft defines some inputs that the browser can look at to determine how to set the various packet markings and defines the a mapping from abstract QoS policies (media type, priority level) to

those packet markings.

2. Relation to Other Standards

This specification does not change or override the advice in any other standards about setting packet markings. It simply provides a non-normative summary of them and provides the context of how they relate into the RTCWeb context. This document also specifies the requirements for the W3C WebRTC API to understand what it needs to control, and how the control splits between things the JavaScript application running in the browser can control and things the browser needs to control. In some cases, such as DSCP where the normative RFC leaves open multiple options to choose from, this clarifies which choice should be used in the RTCWeb context.

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 first input is the type of the media. The browser provides this input as it knows if the media is audio, video, or data. In this specification, both interactive and streaming media is included. They are treated in different categories as their QoS requirements are slightly different. The second input is the relative treatment of the stream within that session. Many applications have multiple video streams and often some are more important than others. JavaScript applications can tell the browser whether a particular media stream is high, medium, or low importance to the application.

5. DSCP Mappings

Below is a table of DSCP markings for each media type RTCWeb is interested in. These DSCPs for each media 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].

Media Type	Low	Medium	High
Audio	46 (EF)	46 (EF)	46 (EF)
Interactive Video	38 (AF43)	36 (AF42)	34 (AF41)
Non-Interactive Video	26 (AF33)	28 (AF32)	30 (AF31)
Data	8 (CS1)	0 (BE)	10 (AF11)

Table 1

6. QCI Mapping

Media Type	Low	Medium	High
Audio	1	1	1
Interactive Video	2	2	2
Non-Interactive Video	8	6	4
Data	9	9	3

Table 2

This corresponds to the mapping provided in TODO REF which are: QCI values (LTE)

Value			Use
1	GBR	2	Interactive Voice
2	GBR	4	Interactive Video
3	GBR	5	Non-Interactive Video
4	GBR	3	Real Time Gaming
5	Non-BG	R 1	IMS Signaling
6	Non-BG	R 7	interactive Voice, video, games
7-9	Non-BG	R 6	non interactive video / TCP web, email, / Platinum vs gold user

Table 3

7. WiFi Mapping

Media Type	Low	Medium	High
Audio	6	6	6
Interactive Video	5	5	5
Non-Interactive Video	4	4	4
Data	1	0	3

Table 4

This corresponds to the mappings from TODO REF of

Value		Traffic Type	Access Category (AC)	Designation
1	BK	Background	AC_BK	Background
2	-	(spare)	AC_BK	Background
0	BE	Best Effort	AC_BE	Best Effort
3	EE	Excellent Effort	AC_BE	Best Effort
4	CL	Controlled Load	AC_VI	Video
5	VI	Video	AC_VI	Video
6	VO	Voice	AC_VO	Voice
7	NC	Network Control	AC_VO	Voice

Table 5

8. W3C API Implications

To work with this proposal, the W3C specification SHOULD provide a way to specify the importance of media and data streams.

The W3C API SHOULD also provide a way for the application to find out the source and destination IP and ports of any flow as well as the DSCP value or other markings in use for that flow. The JavaScript application can then communicate this to a web service that may install a particular policy for that flow.

The W3C API SHOULD NOT provide a way for the JavaScript to arbitrarily set the marketing to any value of the JavaScript choosing as this reduces the security provided by the browser knowing the media type.

9. Security Considerations

TODO - discuss implications of what browser can set and what JavaScript can set

10. IANA Considerations

This specification does not require any actions from IANA.

11. Downward References

This specification contains a downwards reference to [RFC4594] however the parts of that RFC used by this specificaiton are suffenteintly stable for this donward reference.

12. Acknowledgements

Cullen Jennings was one of the authors of this text in the original individual submission but was unceremoniously kicked off by the chairs when it became a WG version. Thanks for hints on code to do this from Paolo Severini, Jim Hasselbrook, Joe Marcus, and Erik Nordmark.

13. 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.

14. Appendix: Code Hints

On windows setting the source interface works but BSD, OSX, Linux use weak end-system model and will route out different interface if that looks like a better route. (TODO - Can someone verify this with specific versions?)

In windows you might be able to tell something about priority of an interface for ICE purposes with WlanQueryInterface or GetIfTable.

The specific mechanisms required to set DSCP code points depend on the application platform.

In windows, setting the DSCP is not easy. See Knowledge Base Article KB248611. TODO - add more information about what can be done for windows.

For most unix variants, the following program can set DSCP.

TODO - make this work in V6. For v6 have a look at IPv6_TCLASS or better the tclass part of sin6_flowid for IPv6

TODO - Can someone test and report back results of program in iOS, Android, Linux, OSX, BSD.

Example test program:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>

#define MSG "Hello, World!"

int
main(void) {
    int sock = -1;
    struct sockaddr *local_addr = NULL;
    struct sockaddr_in sockin, host;
    int tos = 0x60; /* CS3 */
    socklen_t socksiz = 0;
    char *buffer = NULL;

    sock = socket(AF_INET, SOCK_DGRAM, 0);
    if (sock < 0) {
        fprintf(stderr, "Error: %s\n", strerror(errno));
        exit(-1);
    }

    memset(&sockin, 0, sizeof(sockin));
```



```
sockin.sin_family = PF_INET;
sockin.sin_addr.s_addr = inet_addr("11.1.1.1");
socksiz = sizeof(sockin);

local_addr = (struct sockaddr *) &sockin;

/* Set ToS/DSCP */
if (setsockopt(sock, IPPROTO_IP, IP_TOS, &tos,
               sizeof(tos)) < 0) {
    fprintf(stderr, "Error setting TOS: %s\n", strerror(errno));
}

/* Bind to a specific local address */
if (bind(sock, local_addr, socksiz) < 0) {
    fprintf(stderr, "Error binding to socket: %s\n", strerror(errno));
    close(sock); sock=-1;
    exit(-1);
}

buffer = (char *) malloc(strlen(MSG) + 1);
if (buffer == NULL) {
    fprintf(stderr, "Error allocating memory: %s\n", strerror(errno));
    close(sock); sock=-1;
    exit(-1);
}
strcpy(buffer, MSG, strlen(MSG) + 1);
memset(&host, 0, sizeof(host));
host.sin_family = PF_INET;
host.sin_addr.s_addr = inet_addr("10.1.1.1");
host.sin_port = htons(12345);

if (sendto(sock, buffer, strlen(buffer), 0,
           (struct sockaddr *) &host, sizeof(host)) < 0) {
    fprintf(stderr, "Error sending message: %s\n", strerror(errno));
    close(sock); sock=-1;
    free(buffer); buffer=NULL;
    exit(-1);
}

free(buffer); buffer=NULL;
close(sock); sock=-1;

return 0;
}
```

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, August 2006.

15.2. Informative References

- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.
- [RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, March 2002.

Authors' Addresses

Subha Dhesikan
Cisco

Email: sdhesika@cisco.com

Dan Druta (editor)
ATT

Email: dd5826@att.com

Paul Jones
Cisco

Email: paulej@packetizer.com

James Polk
Cisco

Email: jmpolk@cisco.com

TSVWG
Internet-Draft
Intended status: Informational
Expires: August 29, 2013

R. Geib, Ed.
Deutsche Telekom
February 25, 2013

DiffServ interconnection classes and practice
draft-geib-tsvwg-diffserv-intercon-02

Abstract

This document proposes a limited set of interconnection QoS PHBs and PHB groups. It further introduces some DiffServ deployment aspects. The proposals made here should be integrated into a revised version of RFC5127.

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 29, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
1.1. Requirements Language	4
2. Terminology	4
3. An Interconnection class and codepoint scheme	5
4. Consolidation of QoS standards by the interconnection codepoint scheme	6
5. MPLS, Ethernet and IP Precedence for aggregated classes	8
6. QoS class name selection	9
7. Allow for DiffServ extendability on MPLS and Ethernet level .	10
8. Acknowledgements	10
9. IANA Considerations	10
10. Security Considerations	10
11. References	10
11.1. Normative References	10
11.2. Informative References	11
Appendix A. Change log	12
Author's Address	12

1. Introduction

This draft proposes a DiffServ interconnection class and codepoint scheme. At least one party of an interconnection often is a network provider. Aggregated DiffServ classes are often deployed within provider networks. To respect this, this draft also contains concepts and current practice relevant for a revised version of RFC5127 [RFC5127]. Its main purpose is to be considered as an input for the latter task.

DiffServ sees deployment in many networks for the time being. As described in the introduction of the draft diffserv problem statement [I-D.polk-tsvwg-diffserv-stds-problem-statement], remarking of packets at domain boundaries is a DiffServ feature. This draft proposes a set of standard QoS classes and codepoints at interconnection points to which and from which locally used classes and codepoints should be mapped. Such a scheme simplifies interconnection negotiations and ensures that end to end class properties remain roughly the same, even if codepoints change.

The proposed Interconnection class and codepoint scheme tries to reflect and consolidate related DiffServ and QoS standardisation efforts outside of the IETF, namely MEF, GSMA and ITU.

IP Precedence has been deprecated when DiffServ was standardised. It is common practice today however to copy the DSCPs "IP Precedence Bits" into MPLS TC or Ethernet P-Bits, whenever possible. This is reflected by the DiffServ codepoint definitions of AF and EF. This practice and its limits deserve to be documented and discussed briefly.

The draft further adds proposes a philosophy how to add or pick aggregated DiffServ classes. The set of available router and traffic management tools to configure and operate DiffServ classes is limited. This should be reflected by class definitions. These may in the end be more related to transport properties than to application requirements. Please interpret transport properties as "congestion aware" and "not congestion aware" rather than TCP or UDP.

Finally, this draft proposes to leave some MPLS TC codepoint space to allow for future DiffServ extensions like ECN/PCN and domain internal classes (network management traffic is a good example for the latter). An example for an internal PHB may be CS6, which some operators use to protect their network internal routing and / or management traffic. This PHB may not be available to transport customer signaling and management traffic. If IETF is interested in this work, a later version may expand on internal PHBs and codepoints.

In addition to the standardisation activities which triggered this work, other authors published RFCs or drafts which may benefit from an interconnection class- and codepoint scheme. RFC 5160 suggests Meta-QoS-Classes to enable deployment of standardised end to end QoS classes [RFC5160]. The authors agree that the proposed interconnection class- and codepoint scheme as well as the idea of standardised end to end. Hence RFC 5160 and this work complement each other. Work on BGP Class of Service Interconnection signaled by BGP [I-D.knoll-idr-cos-interconnect] is beyond the the scope of this draft. Should the basic transport and class properties of end to end QoS utilising DiffServ based interconnection as proposed by this draft be standardised, work on signaled access to QoS classes may be of interest.

1.1. 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 RFC 2119 [RFC2119].

2. Terminology

This draft tries to re-use existing terminology. It further tries to indicate, where duplicate terminology may exist.

Class A class is a set of one or more PHBs. If a class consists of a set of PHBs and these obey to an ordering constraint. In that sense, a class is a single AF class (e.g. AF4 consisting of AF41, AF42 and AF43) [RFC2597]. A class is a PHB group [RFC2575] and a PHB scheduling class [RFC3260]. On IP level all DSCPs sharing the same IP precedence value belong to a single class. A class may consist of one or more PHBs. A single class uses forwarding resources, which are independent of the forwarding resources of any other class. Different classes must not be aggregated.

PHB A single Per Hop Behaviour [RFC2575] is identified by a single DSCP on IP layer.

Many DiffServ related RFCs introduce new terminology duplicating the existing one. The above references are incomplete and refer to the early DiffServ RFCs only. Stopping terminology duplication may simplify discussion.

The following current practice issues relate to the concept of the DiffServ interconnection class proposal rather than to terminology. They serve as additional motivation of this activity:

- o Abstract class names like "EF" are preferential over those being close to an application, like "Voice". Unfortunately, non QoS experts can't handle abstract class names. Hence and usually sooner than later, classes are named for applications or groups of them. One consequence however is, that people tend to combine application group class names and SLA parameters. Based on an application specific name and some worst case performance numbers on a paper, they often decide that their application needs a separate new QoS class.
- o Worse than that, but very present in practice, is the class abstraction level which is preferred by those dealing with QoS (as experts or non experts): the DSCPs or the IP precedence values. These are the commodity abstractions applied for QoS classes. Most of these persons have fixed class to codepoint mappings in their minds, which they can't easily adapt on a per customer or interconnection partner basis.

While these issues aren't to be solved by IETF (QoS experts could and should of course teach staff to use proper DiffServ terminology and concepts), a simple and comprehensible QoS interconnection class scheme also is helpful in this area.

3. An Interconnection class and codepoint scheme

DiffServ deployments mostly follow loose class specification schemes (often one or two AF classes, EF and Best Effort). Especially DSCP assignment for the AF classes varies between deployments. Basic AF class definitions are often similar however. This is in line with the DiffServ architecture. This document doesn't propose to change that.

Interconnecting parties face the problem of matching classes to be interconnected and then to agree on codepoint mapping. As stated by draft diffserv-problem-statement [I-D.polk-tsvwg-diffserv-stds-problem-statement], remarking is a standard behaviour at interconnection interfaces. This draft proposes a set of 4 QoS classes with a set of well defined DSCPs and IP-Precedence values as interconnection class and codepoint scheme. A sending party remarks DSCPs from internal schemes to the Interconnection codepoints. The receiving party remarks IP-Precedence and or DSCPs to their internal scheme. Thus the interconnection codepoint scheme fully complies with the DiffServ architecture. Such an interconnection class and codepoint scheme was introduced by ITU-T [Y.1566] (there also including Ethernet). It is specified to a higher level of detail in this document.

At first glance, this looks like an additional effort. But there are obvious benefits: each party sending or receiving traffic has to specify the mapping from or to the interconnection class and codepoint scheme only once. Without it, this is to be negotiated per interconnection party individually. Further, end-to-end QoS in terms of traffic being classified for the same class in all passed domains is likely to result if an interconnection codepoint scheme is used. It is not necessarily resulting from individual per network mapping negotiations.

The standards and deployments known to the author of this draft are limited to 4 DiffServ classes at interconnection points (or less). Draft RFC 4597 update [I-D.polk-tsvwg-rfc4594-update] doesn't seem to generally contradict to this, as it proposes to standardise "many services classes, not all will be used in each network at any period of time." Some more good reasons favour working with 4 DiffServ interconnection classes for now:

- o There should be a coding reserve for interconnection classes, leaving space for future standards, for bilateral agreements and for carrier internal classes.
- o MPLS and Ethernet support only 8 PHBs, classes or ECN indications. Assignment of codepoints for whatever purpose must be well thought through. Limiting interconnection QoS to four classes is MPLS and Ethernet friendly in that sense.
- o Migrations from one codepoint scheme to another may require spare QoS codepoints.

4. Consolidation of QoS standards by the interconnection codepoint scheme

The interconnection class and codepoint scheme proposed by Y.1566 also tries to consolidate related DiffServ and QoS standardisation efforts outside of the IETF [Y.1566]. The interconnection class and codepoint scheme may be a suitable approach to consolidate these standards. MEF 23.1 specifies 3 aggregated classes, consuming up to 5 codepoints on Ethernet layer (EF, AF3 and AF1 and Best Effort) and 6 PHBs [MEF23.1]. MEF aggregates AF1 and Default PHB in a single class. This is not recommended for interconnection, as it is not in line with RFC 2597 (which requires separate forwarding resources for each AF class and doesn't foresee aggregation of Default PHB and an AF class).

GSMA IR.34 proposes four classes, EF, AF4, another AF class and Best Effort with 7 PHBs in sum [IR.34]. IR.34 specifies an "Interactive"

class consisting of 3 PHBs with different drop priorities. IR.34 specifies the PHBs AF31, AF21 and AF11 for this Interactive class. This definitely breaks RFC 2597. The interconnection class and codepoint scheme supports the Interactive class but assigns AF3 with PHBs AF31, AF32 and AF33.

If IETF picks up this draft, it may be a good idea to inform MEF and GSMA about conflicts of their standards with DiffServ and suggest joint activities to improve the situation. Information on interworkings with MEF 23 and GSMA IR.34 with the interconnection QoS scheme could be given by a later version of this draft.

The classes to be supported at interconnection interfaces are specified by Y.1566 as:

Class Priority: EF, expecting the figures of merit describing the PHB to be in the range of low single digit milliseconds. See [RFC3246].

Bulk inelastic: Optimised for low loss, low delay, low jitter at high bandwidth. Traffic load in this class must be controlled, e.g. by application servers. One example could be flow admission control. There may be infrequent retransmissions requested by the application layer to mitigate low levels of packet losses. Discard of packets through active queue management should be avoided in this class. Congestion in this class may result in bursty packet loss. If used to carry multimedia traffic, it is recommended to carry audio and video traffic in a single PHB. All of these properties influence the buffer design.

Assured: This class may be optimised to transport traffic without bandwidth requirements. It aims on very low loss at high bandwidths. Retransmissions after losses characterise the class and influence the buffer design. Active queue management with probabilistic dropping may be deployed.

Default: Default. This class may be optimised to transport traffic without bandwidth requirements. Retransmissions after losses characterise the class and influence the buffer design. Active queue management with probabilistic dropping may be deployed.

Note that other DiffServ related standards trim down class requirements to SLA parameters. To quote e.g RFC 4594-update, "A "service class" represents a similar set of traffic characteristics for delay, loss, and jitter as packets traverse routers in a network." This draft adds traffic conditioning properties

corresponding to expected transport layer characteristics as a key factor to a class definition: the desired class performance like delay, jitter and worst case loss are met only if conditioning and transport properties meet the ones described by the class definition. This is not to say, the other standards ignore conditioner properties. They are e.g. a core part of RFC 4594-update. They do not directly refer to transport protocol properties, as most existing QoS standards prefer the approach of assigning QoS classes to applications or application sets. This may result in undesirable class mappings, if an e.g. IP TV application demanding low loss is matched to a class whose low loss guarantees depend on AQM mechanisms.

Y.1566 does not recommend all PHBs to be supported at an interconnection interface. This information is added by this draft. At interconnection points, the following PHBs should be accepted between interconnected parties:

Class: PHB (one or more)

Class Priority: EF

Bulk inelastic: AF41 (AF42 and AF43 are reserved for extension)

Assured: AF31, AF32 and AF33

Default: Default

Class names (and property specification) are picked from Y.1566. PHBs to the level of detail introduced here are not part of Y.1566.

5. MPLS, Ethernet and IP Precedence for aggregated classes

IP Precedence has been deprecated when DiffServ was standardised. Ethernet and MPLS support 3 bit codepoint fields to differentiate service quality. Mapping of the IP precedence to these 3 Bit fields has been a configuration restriction in the early days of DiffServ. The concept of paying attention to the three most significant bits of a DSCP has however been part of Diffserv from start on (EF's IP Precedence is 5, that of AF4 is 4 and so on). The interconnection class and codepoint scheme respects this in different ways:

- o it allows to classify four interconnection classes based on IP precedence.
- o It supports a single PHB group (AF3), which may be mapped to up to three different MPLS TC's or Ethernet P-Bits. Note that this

draft doesn't favour or recommend doing that, but it is possible. The author isn't aware of deployed service offers with 3 different drop levels in a single class.

This is of course no requirement to deprecate any DSCP to MPLS TC or Ethernet P-Bit mapping functionality. This functionality is very important as well.

6. QoS class name selection

This is more of an informational discussion, proposed best practice, and mainly relates to human behaviour (including QoS experts) rather than technical issues. Above the human preference for conceivable class names has been mentioned. Network engineers (including the former Diffserv WG authors) recommend to avoid application related QoS class names. Focus should be put on class properties. But these can be irritating again, as just looking at SLA parameters like Delay, Jitter and packet loss don't tell the reader, which conditioning and transport properties guided the class engineering assumptions resulted in the conditioning of a class. A router produces QoS with a scheduling mechanism, a settable queue depth and optional active queue management (including ECN), and may be a policer. Some kind of resource management may be present (also in Diffserv domains). It's beyond the imagination of the author how one would engineer more than half a dozen classes with distinguishable properties with this set of tools.

There's no perfect solution to the problem, as conditioning configurations are not comprehensible to most readers, even if they were communicated (they are operational secrets of course). There are (or should be) engineering assumptions, when designing QoS conditioners. But they closer relate to layer 3 or layer 4 level properties than to specific applications. In general, an application responds to congestion by reducing traffic, or it ignores congestion. Active queue management doesn't help to avoid congestion in the latter case, only resource management does. EF may be a special case. If the EF traffic is not responsive to congestion, and packets are assumed to be short, rather small jitter values can be reached if engineering ensures that the packet arrival rate never exceeds the transmission rate of that queue (see RFC 3246 [RFC3246]). There's other non congestion-responsive traffic, for which the EF engineering assumptions may not fit. So conditioning like bulk inelastic is reasonable.

Active queue management may be deployed for QoS classes, which are designed to transport traffic responding to congestion by traffic reduction.

The class names of this document follow Y.1566. TCP_optimised and especially UDP_optimised are inappropriate as class names, as some UDP based application are or may be expected to become TCP friendly.

7. Allow for DiffServ extendability on MPLS and Ethernet level

Any aggregated Diffserv deployment faces codepoint depletion issues rather soon, if deployed on MPLS or Ethernet. Coding space should be left for new features, like ECN, PCN or Conex. In addition to carrying customer traffic, internal routing and network management traffic may be protected by using a separate class. Offering interconnection with up to four classes and 4 - 6 MPLS TC's (or Ethernet P-bits) to that respect is probably at least a fair compromise.

8. Acknowledgements

David Black gave many helpful comments to this work. Al Morton and Sebastien Jobert provided feedback on many aspects during private discussions. Brian Carpenter, Mohamed Boucadair and Thomas Knoll helped adding awareness of further potentially related work.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

This document does not introduce new features, it describes how to use existing ones. The security section of RFC 4597 [RFC4597] applies.

11. References

11.1. Normative References

- [RFC2575] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2575, April 1999.
- [RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.

- [RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, March 2002.
- [RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, April 2002.
- [min_ref] authSurName, authInitials., "Minimal Reference", 2006.

11.2. Informative References

- [I-D.knoll-idr-cos-interconnect]
Knoll, T., "BGP Class of Service Interconnection",
draft-knoll-idr-cos-interconnect-09 (work in progress),
November 2012.
- [I-D.polk-tsvwg-diffserv-stds-problem-statement]
Polk, J., "The Problem Statement for the Standard
Configuration of DiffServ Service Classes",
draft-polk-tsvwg-diffserv-stds-problem-statement-00 (work
in progress), July 2012.
- [I-D.polk-tsvwg-rfc4594-update]
Polk, J., "Standard Configuration of DiffServ Service
Classes", draft-polk-tsvwg-rfc4594-update-02 (work in
progress), October 2012.
- [IR.34] GSMA Association, "IR.34 Inter-Service Provider IP
Backbone Guidelines Version 7.0", GSMA, GSMA IR.34 [http://
www.gsma.com/newsroom/wp-content/uploads/2012/03/
ir.34.pdf](http://www.gsma.com/newsroom/wp-content/uploads/2012/03/ir.34.pdf), 2012.
- [MEF23.1] MEF, "Implementation Agreement MEF 23.1 Carrier Ethernet
Class of Service Phase 2", MEF, MEF23.1 [http://
metroethernetforum.org/PDF_Documents/
technical-specifications/MEF_23.1.pdf](http://metroethernetforum.org/PDF_Documents/technical-specifications/MEF_23.1.pdf), 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4597] Even, R. and N. Ismail, "Conferencing Scenarios",
RFC 4597, August 2006.
- [RFC5127] Chan, K., Babiarz, J., and F. Baker, "Aggregation of
Diffserv Service Classes", RFC 5127, February 2008.
- [RFC5160] Levis, P. and M. Boucadair, "Considerations of Provider-

to-Provider Agreements for Internet-Scale Quality of Service (QoS)", RFC 5160, March 2008.

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

Appendix A. 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.

Author's Address

Ruediger Geib (editor)
Deutsche Telekom
Heinrich Hertz Str. 3-7
Darmstadt, 64297
Germany

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

Network Working Group
Internet-Draft
Updates: 2460 (if approved)
Intended status: Standards Track
Expires: August 25, 2013

M. Eubanks
AmericaFree.TV LLC
P. Chimento
Johns Hopkins University Applied
Physics Laboratory
M. Westerlund
Ericsson
February 21, 2013

IPv6 and UDP Checksums for Tunneled Packets
draft-ietf-6man-udpchecksums-08

Abstract

This document provides an update of the Internet Protocol version 6 (IPv6) specification (RFC2460) to improve the performance in the use case where a tunnel protocol uses UDP with IPv6 to tunnel packets. The performance improvement is obtained by relaxing the IPv6 UDP checksum requirement for any suitable tunnel protocol where header information is protected on the "inner" packet being carried. This relaxation removes the overhead associated with the computation of UDP checksums on IPv6 packets used to carry tunnel protocols. The specification describes how the IPv6 UDP checksum requirement can be relaxed for the situation where the encapsulated packet itself contains a checksum. The limitations and risks of this approach are described, and restrictions specified on the use of the method.

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 25, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the

document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Some Terminology	4
2.1. Requirements Language	4
3. Problem Statement	4
4. Discussion	4
4.1. Analysis of Corruption in Tunnel Context	5
4.2. Limitation to Tunnel Protocols	7
4.3. Middleboxes	8
5. The Zero-Checksum Update	8
6. Additional Observations	10
7. IANA Considerations	10
8. Security Considerations	10
9. Acknowledgements	11
10. References	11
10.1. Normative References	11
10.2. Informative References	12
Authors' Addresses	12

1. Introduction

This work constitutes an update of the Internet Protocol Version 6 (IPv6) Specification [RFC2460], in the use case where a tunnel protocol uses UDP with IPv6 to tunnel packets. With the rapid growth of the Internet, tunnel protocols have become increasingly important to enable the deployment of new protocols. Tunnel protocols can be deployed rapidly, while the time to upgrade and deploy a critical mass of routers, middleboxes and hosts on the global Internet for a new protocol is now measured in decades. At the same time, the increasing use of firewalls and other security-related middleboxes means that truly new tunnel protocols, with new protocol numbers, are also unlikely to be deployable in a reasonable time frame, which has resulted in an increasing interest in and use of UDP-based tunnel protocols. In such protocols, there is an encapsulated "inner" packet, and the "outer" packet carrying the tunneled inner packet is a UDP packet, which can pass through firewalls and other middleboxes that perform filtering that is a fact of life on the current Internet.

Tunnel endpoints may be routers or middleboxes aggregating traffic from a number of tunnel users, therefore the computation of an additional checksum on the outer UDP packet may be seen as an unwarranted burden on nodes that implement a tunnel protocol, especially if the inner packet(s) are already protected by a checksum. In IPv4, there is a checksum over the IP packet header, and the checksum on the outer UDP packet may be set to zero. However in IPv6 there is no checksum in the IP header and RFC 2460 [RFC2460] explicitly states that IPv6 receivers MUST discard UDP packets with a zero checksum. So, while sending a UDP datagram with a zero checksum is permitted in IPv4 packets, it is explicitly forbidden in IPv6 packets. To improve support for IPv6 UDP tunnels, this document updates RFC 2460 to allow endpoints to use a zero UDP checksum under constrained situations (primarily IPv6 tunnel transports that carry checksum-protected packets), following the applicability statements and constraints in [I-D.ietf-6man-udpzero].

"Unicast UDP Usage Guidelines for Application Designers" [RFC5405] should be consulted when reading this specification. It discusses both UDP tunnels (Section 3.1.3) and the usage of checksums (Section 3.4).

While the origin of this specification is the problem raised by the draft titled "Automatic Multicast Tunnels", also known as "AMT" [I-D.ietf-mboned-auto-multicast] we expect it to have wide applicability. Since the first version of this document, the need for an efficient UDP tunneling mechanism has increased. Other IETF Working Groups, notably LISP [RFC6830] and Softwires [RFC5619] have

expressed a need to update the UDP checksum processing in RFC 2460. We therefore expect this update to be applicable in the future to other tunnel protocols specified by these and other IETF Working Groups.

2. Some Terminology

This document discusses only IPv6, since this problem does not exist for IPv4. Therefore all reference to 'IP' should be understood as a reference to IPv6.

The document uses the terms "tunneling" and "tunneled" as adjectives when describing packets. When we refer to 'tunneling packets' we refer to the outer packet header that provides the tunneling function. When we refer to 'tunneled packets' we refer to the inner packet, i.e., the packet being carried in the tunnel.

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 RFC 2119 [RFC2119].

3. Problem Statement

When using tunnel protocols based on UDP, there can be both a benefit and a cost to computing and checking the UDP checksum of the outer (encapsulating) UDP transport header. In certain cases, reducing the forwarding cost is important, e.g., for nodes that perform the checksum in software the cost may outweigh the benefit. This document provides an update for usage of the UDP checksum with IPv6. The update is specified for use by a tunnel protocol that transports packets that are themselves protected by a checksum.

4. Discussion

"Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero] describes issues related to allowing UDP over IPv6 to have a valid zero UDP checksum and is the starting point for this discussion. Sections 4 and 5 of [I-D.ietf-6man-udpzero], respectively identify node implementation and usage requirements for datagrams sent and received with a zero UDP checksum. These introduce constraints on the usage of a zero checksum for UDP over IPv6. The remainder of this section analyses the use of general tunnels and motivates why tunnel protocols are

being permitted to use the method described in this update. Issues with middleboxes are also discussed.

4.1. Analysis of Corruption in Tunnel Context

This section analyzes the impact of the different corruption modes in the context of a tunnel protocol. It indicates what needs to be considered by the designer and user of a tunnel protocol to be robust. It also summarizes why use of a zero UDP checksum is thought to be safe for deployment.

1. Context (i.e., tunneling state) should be established by exchanging application Protocol Data Units (PDUs) carried in checksummed UDP datagrams or by other protocols with integrity protection against corruption. These control packets should also carry any negotiation required to enable the tunnel endpoint to accept UDP datagrams with a zero checksum and identify the set of ports that are used. It is important that the control traffic is robust against corruption because undetected errors can lead to long-lived and significant failures that may affect much more than the single packet that was corrupted.
2. Keep-alive datagrams with a zero UDP checksum should be sent to validate the network path, because the path between tunnel endpoints can change and therefore the set of middleboxes along the path may change during the life of an association. Paths with middleboxes that drop datagrams with a zero UDP checksum will drop these keep-alives. To enable the tunnel endpoints to discover and react to this behavior in a timely way, the keep-alive traffic should include datagrams with a non-zero checksum and datagrams with a zero checksum.
3. Receivers should attempt to detect corruption of the address information in an encapsulating packet. A robust tunnel protocol should track tunnel context based on the 5-tuple (tunneled protocol number, IPv6 source address, IPv6 destination address, UDP source port, UDP destination port). A corrupted datagram that arrives at a destination may be filtered based on this check.
 - * If the datagram header matches the 5-tuple and the node has the zero checksum enabled for this port, the payload is matched to the wrong context. The tunneled packet will then be decapsulated and forwarded by the tunnel egress.
 - * If a corrupted datagram matches a different 5-tuple and the zero checksum was enabled for the port, the datagram payload is matched to the wrong context, and may be processed by the

wrong tunnel protocol, if it also passes the verification of that protocol.

- * If a corrupted datagram matches a 5-tuple and the zero checksum has not been enabled for this port, the datagram will be discarded.

When only the source information is corrupted, the datagram could arrive at the intended applications/protocol, which will process the datagram and try to match it against an existing tunnel context. The likelihood that a corrupted packet enters a valid context is reduced when the protocol restricts processing to only the source addresses with established contexts. When both source and destination fields are corrupted, this increases the likelihood of failing to match a context, with the exception of errors replacing one packet header with another one. In this case, it is possible that both packets are tunneled and therefore the corrupted packet could match a previously defined context.

4. Receivers should attempt to detect corruption of source-fragmented encapsulating packets. A tunnel protocol may reassemble fragments associated with the wrong context at the right tunnel endpoint, or it may reassemble fragments associated with a context at the wrong tunnel endpoint, or corrupted fragments may be reassembled at the right context at the right tunnel endpoint. In each of these cases, the IPv6 length of the encapsulating header may be checked (though [I-D.ietf-6man-udpzero] points out the weakness in this check). In addition, if the encapsulated packet is protected by a transport (or other) checksum, these errors can be detected (with some probability).
5. Tunnel protocols using UDP have some advantages that reduce the risk for a corrupted tunnel packet reaching a destination that will receive it, compared to other applications. This results from processing by the network of the inner (tunneled) packet after being forwarded from the tunnel egress using a wrong context:
 - * A tunneled packet may be forwarded to the wrong address domain, for example, a private address domain where the inner packet's address is not routable, or may fail a source address check, such as Unicast Reverse Path Forwarding [RFC2827], resulting in the packet being dropped.
 - * The destination address of a tunneled packet may not at all be reachable from the delivered domain. For example, an Ethernet

frame where the destination MAC address is not present on the LAN segment that was reached.

- * The type of the tunneled packet may prevent delivery. For example, an attempt to interpret an IP packet payload as an Ethernet frame, would likely to result in the packet being dropped as invalid.
- * The tunneled packet checksum or integrity mechanism may detect corruption of the inner packet caused at the same time as corruption to the outer packet header. The resulting packet would likely be dropped as invalid.

These checks each significantly reduce the likelihood that a corrupted inner tunneled packet is finally delivered to a protocol listener that can be affected by the packet. While the methods do not guarantee correctness, they can reduce the risk of relaxing the UDP checksum requirement for a tunnel application using IPv6.

4.2. Limitation to Tunnel Protocols

This document describes the applicability of using a zero UDP checksum to support tunnel protocols. There are good motivations behind this and the arguments are provided here.

- o Tunnels carry inner packets that have their own semantics, which may make any corruption less likely to reach the indicated destination and be accepted as a valid packet. This is true for IP packets with the addition of verification that can be made by the tunnel protocol, the network processing of the inner packet headers as discussed above, and verification of the inner packet checksums. Non-IP inner packets are likely to be subject to similar effects that may reduce the likelihood of a misdelivered packet being delivered to a protocol listener that can be affected by the packet.
- o Protocols that directly consume the payload must have sufficient robustness against misdelivered packets from any context, including the ones that are corrupted in tunnels and any other usage of the zero checksum. This will require an integrity mechanism. Using a standard UDP checksum reduces the computational load in the receiver to verify this mechanism.
- o The design for stateful protocols or protocols where corruption causes cascade effects requires extra care. In tunnel usage, each encapsulating packet provides only a transport mechanism from tunnel ingress to tunnel egress. A corruption will commonly only affect the single tunneled packet, not the established protocol

state. One common effect is that the inner packet flow will only see a corruption and misdelivery of the outer packet as a lost packet.

- o Some non-tunnel protocols operate with general servers that do not know the source from which they will receive a packet. In such applications, a zero UDP checksum is unsuitable because there is a need to provide the first level of verification that the packet was intended for the receiving server. A verification prevents the server from processing the datagram payload and without this it may spend significant resources processing the packet, including sending replies or error messages.

Tunnel protocols that encapsulate IP will generally be safe for deployment, since all IPv4 and IPv6 packets include at least one checksum at either the network or transport layer. The network delivery of the inner packet will then further reduce the effects of corruption. Tunnel protocols carrying non-IP packets may offer equivalent protection when the non-IP networks reduce the risk of misdelivery to applications. However, there is a need for further analysis to understand the implications of misdelivery of corrupted packets for that each non-IP protocol. The analysis above suggests that non-tunnel protocols can be expected to have significantly more cases where a zero checksum would result in misdelivery or negative side-effects.

One unfortunate side-effect of increased use of a zero-checksum is that it also increases the likelihood of acceptance when a datagram with a zero UDP checksum is misdelivered. This requires all tunnel protocols using this method to be designed to be robust to misdelivery.

4.3. Middleboxes

"Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero] notes that middleboxes that conform to RFC 2460 will discard datagrams with a zero UDP checksum and should log this as an error. Tunnel protocols intending to use a zero UDP checksum need to ensure that they have defined a method for handling cases when a middlebox prevents the path between the tunnel ingress and egress from supporting transmission of datagrams with a zero UDP checksum.

5. The Zero-Checksum Update

This specification updates IPv6 to allow a zero UDP checksum in the outer encapsulating datagram of a tunnel protocol. UDP endpoints

that implement this update MUST follow the node requirements in "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].

The following text in [RFC2460] Section 8.1, 4th bullet should be deleted:

"Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error."

This text should be replaced by:

An IPv6 node associates a mode with each used UDP port (for sending and/or receiving packets).

Whenever originating a UDP packet for a port in the default mode, an IPv6 node MUST compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it MUST be changed to hex FFFF for placement in the UDP header as specified in [RFC2460]. IPv6 receivers MUST by default discard UDP packets containing a zero checksum, and SHOULD log the error.

As an alternative, certain protocols that use UDP as a tunnel encapsulation, MAY enable the zero-checksum mode for a specific port (or set of ports) for sending and/or receiving. Any node implementing the zero-checksum mode MUST follow the node requirements specified in Section 4 of "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].

Any protocol that enables the zero-checksum mode for a specific port or ports MUST follow the usage requirements specified in Section 5 of "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].

Middleboxes supporting IPv6 MUST follow requirements 9, 10 and 11 of the usage requirements specified in Section 5 of "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].

6. Additional Observations

This update was motivated by the existence of a number of protocols being developed in the IETF that are expected to benefit from the change. The following observations are made:

- o An empirically-based analysis of the probabilities of packet corruption (with or without checksums) has not (to our knowledge) been conducted since about 2000. At the time of publication, it is now 2012. We strongly suggest a new empirical study, along with an extensive analysis of the corruption probabilities of the IPv6 header. This can potentially allow revising the recommendations in this document.
- o A key motivation for the increase in use of UDP in tunneling is a lack of protocol support in middleboxes. Specifically, new protocols, such as LISP [RFC6830], may prefer to use UDP tunnels to traverse an end-to-end path successfully and avoid having their packets dropped by middleboxes. If middleboxes were updated to support UDP-Lite [RFC3828], UDP-Lite would provide better protection than offered by this update. This may be suited to a variety of applications and would be expected to be preferred over this method for many tunnel protocols.
- o Another issue is that the UDP checksum is overloaded with the task of protecting the IPv6 header for UDP flows (as is the TCP checksum for TCP flows). Protocols that do not use a pseudo-header approach to computing a checksum or CRC have essentially no protection from misdelivered packets.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

Less work is required to generate an attack using a zero UDP checksum than one using a standard full UDP checksum. However, this does not lead to significant new vulnerabilities because checksums are not a security measure and can be easily generated by any attacker.

In general any user of zero UDP checksums should apply the checks and context verification that are possible to minimize the risk of

unintended traffic to reach a particular context. This will however not protect against an intended attack that create packet with the correct information. Source address validation can help prevent injection of traffic into contexts by an attacker.

Depending on the hardware design, the processing requirements may differ for tunnels that have a zero UDP checksum and those that calculate a checksum. This processing overhead may need to be considered when deciding whether to enable a tunnel and to determine an acceptable rate for transmission. This can become a security risk for designs that can handle a significantly larger number of packets with zero UDP checksums compared to datagrams with a non-zero checksum, such as tunnel egress. An attacker could attempt to inject non-zero checksummed UDP packets into a tunnel forwarding zero checksum UDP packets and cause overload in the processing of the non-zero checksums, e.g. if this happens in a routers slow path. Protection mechanisms should therefore be employed when this threat exists. Protection may include source address filtering to prevent an attacker injecting traffic, as well as throttling the amount of non-zero checksum traffic. The latter may impact the function of the tunnel protocol.

9. Acknowledgements

We would like to thank Brian Haberman, Dan Wing, Joel Halpern, David Waltermire, J.W. Atwood, Peter Yee, Joe Touch and the IESG of 2012 for discussions and reviews. Gorrry Fairhurst has been very diligent in reviewing and help ensuring alignment between this document and [I-D.ietf-6man-udpzero].

10. References

10.1. Normative References

- [I-D.ietf-6man-udpzero]
Fairhurst, G. and M. Westerlund, "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums", draft-ietf-6man-udpzero-10 (work in progress), January 2013.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

10.2. Informative References

- [I-D.ietf-mboned-auto-multicast]
Bumgardner, G., "Automatic Multicast Tunneling",
draft-ietf-mboned-auto-multicast-14 (work in progress),
June 2012.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering:
Defeating Denial of Service Attacks which employ IP Source
Address Spoofing", BCP 38, RFC 2827, May 2000.
- [RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and
G. Fairhurst, "The Lightweight User Datagram Protocol
(UDP-Lite)", RFC 3828, July 2004.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines
for Application Designers", BCP 145, RFC 5405,
November 2008.
- [RFC5619] Yamamoto, S., Williams, C., Yokota, H., and F. Parent,
"Software Security Analysis and Requirements", RFC 5619,
August 2009.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The
Locator/ID Separation Protocol (LISP)", RFC 6830,
January 2013.

Authors' Addresses

Marshall Eubanks
AmericaFree.TV LLC
P.O. Box 141
Clifton, Virginia 20124
USA

Phone: +1-703-501-4376
Fax:
Email: marshall.eubanks@gmail.com

P.F. Chimento
Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723
USA

Phone: +1-443-778-1743
Email: Philip.Chimento@jhuapl.edu

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2013

G. Fairhurst
University of Aberdeen
M. Westerlund
Ericsson
February 25, 2013

Applicability Statement for the use of IPv6 UDP Datagrams with Zero
Checksums
draft-ietf-6man-udpzero-12

Abstract

This document provides an applicability statement for the use of UDP transport checksums with IPv6. It defines recommendations and requirements for the use of IPv6 UDP datagrams with a zero UDP checksum. It describes the issues and design principles that need to be considered when UDP is used with IPv6 to support tunnel encapsulations and examines the role of the IPv6 UDP transport checksum. The document also identifies issues and constraints for deployment on network paths that include middleboxes. An appendix presents a summary of the trade-offs that were considered in evaluating the safety of the update to RFC 2460 that updates use of the UDP checksum with IPv6.

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 29, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

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

Table of Contents

1. Introduction	4
1.1. Document Structure	5
1.2. Terminology	5
1.3. Use of UDP Tunnels	5
1.3.1. Motivation for new approaches	6
1.3.2. Reducing forwarding cost	6
1.3.3. Need to inspect the entire packet	7
1.3.4. Interactions with middleboxes	7
1.3.5. Support for load balancing	8
2. Standards-Track Transports	9
2.1. UDP with Standard Checksum	9
2.2. UDP-Lite	9
2.2.1. Using UDP-Lite as a Tunnel Encapsulation	10
2.3. General Tunnel Encapsulations	10
2.4. Relation to UDP-Lite and UDP with checksum	10
3. Issues Requiring Consideration	12
3.1. Effect of packet modification in the network	13
3.1.1. Corruption of the destination IP address	14
3.1.2. Corruption of the source IP address	15
3.1.3. Corruption of Port Information	16
3.1.4. Delivery to an unexpected port	16
3.1.5. Corruption of Fragmentation Information	17
3.2. Where Packet Corruption Occurs	19
3.3. Validating the network path	20
3.4. Applicability of method	21
3.5. Impact on non-supporting devices or applications	21
4. Constraints on implementation of IPv6 nodes supporting zero checksum	22
5. Requirements on usage of the zero UDP checksum	24
6. Summary	26
7. Acknowledgements	28
8. IANA Considerations	28
9. Security Considerations	28
10. References	29
10.1. Normative References	29
10.2. Informative References	29

Appendix A. Evaluation of proposal to update RFC 2460 to support zero checksum	31
A.1. Alternatives to the Standard Checksum	31
A.2. Comparison	33
A.2.1. Middlebox Traversal	33
A.2.2. Load Balancing	34
A.2.3. Ingress and Egress Performance Implications	34
A.2.4. Deployability	34
A.2.5. Corruption Detection Strength	35
A.2.6. Comparison Summary	35
Appendix B. Document Change History	38
Authors' Addresses	41

1. Introduction

The User Datagram Protocol (UDP) [RFC0768] transport is defined for the Internet Protocol (IPv4) [RFC0791] and is defined in "Internet Protocol, Version 6 (IPv6) [RFC2460] for IPv6 hosts and routers. The UDP transport protocol has a minimal set of features. This limited set has enabled a wide range of applications to use UDP, but these application do need to provide many important transport functions on top of UDP. The UDP Usage Guidelines [RFC5405] provides overall guidance for application designers, including the use of UDP to support tunneling. The key difference between UDP usage with IPv4 and IPv6 is that RFC 2460 mandates use of a calculated UDP checksum, i.e. a non-zero value, due to the lack of an IPv6 header checksum. The inclusion of the pseudo header in the checksum computation provides a statistical check that datagrams have been delivered to the intended IPv6 destination node. Algorithms for checksum computation are described in [RFC1071].

The lack of a possibility to use an IPv6 datagram with a zero UDP checksum has been observed as a real problem for certain classes of application, primarily tunnel applications. This class of application has been deployed with a zero UDP checksum using IPv4. The design of IPv6 raises different issues when considering the safety of using a UDP checksum with IPv6. These issues can significantly affect applications, both when an endpoint is the intended user and when an innocent bystander (when a packet is received by a different endpoint to that intended).

This document examines the issues and an appendix compares the strengths and weaknesses of a number of proposed solutions. This identifies a set of issues that must be considered and mitigated to be able to safely deploy IPv6 applications that use a zero UDP checksum. The provided comparison of methods is expected to also be useful when considering applications that have different goals from the ones that initiated the writing of this document, especially the use of already standardized methods. The analysis concludes that using a zero UDP checksum is the best method of the proposed alternatives to meet the goals for certain tunnel applications.

This document defines recommendations and requirements for use of IPv6 datagrams with a zero UDP checksum. This usage is expected to have initial deployment issues related to middleboxes, limiting the usability more than desired in the currently deployed Internet. However, this limitation will be largest initially and will reduce as updates are provided in middleboxes that support the zero UDP checksum for IPv6. The document therefore derives a set of constraints required to ensure safe deployment of a zero UDP checksum.

Finally, the document also identifies some issues that require future consideration and possibly additional research.

1.1. Document Structure

Section 1 provides a background to key issues, and introduces the use of UDP as a tunnel transport protocol.

Section 2 describes a set of standards-track datagram transport protocols that may be used to support tunnels.

Section 3 discusses issues with a zero UDP checksum for IPv6. It considers the impact of corruption, the need for validation of the path and when it is suitable to use a zero UDP checksum.

Section 4 is an applicability statement that defines requirements and recommendations on the implementation of IPv6 nodes that support the use of a zero UDP checksum.

Section 5 provides an applicability statement that defines requirements and recommendations for protocols and tunnel encapsulations that are transported over an IPv6 transport that does not perform a UDP checksum calculation to verify the integrity at the transport endpoints.

Section 6 provides the recommendations for standardization of zero UDP checksum with a summary of the findings and notes remaining issues needing future work.

Appendix A evaluates the set of proposals to update the UDP transport behaviour and other alternatives intended to improve support for tunnel protocols. It concludes by assessing the trade-offs of the various methods, identifying advantages and disadvantages for each method.

1.2. Terminology

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

1.3. Use of 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. Using UDP as a tunneling protocol is attractive when the payload protocol is not supported by the middleboxes that may exist along the path,

because many middleboxes support transmission using UDP. In this use, the receiving endpoint decapsulates the UDP datagrams and forwards the original packets contained in the payload [RFC5405]. 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.

1.3.1. Motivation for new approaches

A number of tunnel encapsulations deployed over IPv4 have used the UDP transport with a zero checksum. Users of these protocols expect a similar solution for IPv6.

A number of tunnel protocols are also currently being defined (e.g. Automated Multicast Tunnels, AMT [I-D.ietf-mboned-auto-multicast], and the Locator/Identifier Separation Protocol, LISP [LISP]). These protocols motivated an update to IPv6 UDP checksum processing to benefit from simpler checksum processing for various reasons:

- o Reducing forwarding costs, motivated by redundancy present in the encapsulated packet header, since in tunnel encapsulations, payload integrity and length verification may be provided by higher layer encapsulations (often using the IPv4, UDP, UDP-Lite, or TCP checksums).
- o Eliminating a need to access the entire packet when forwarding the packet by a tunnel endpoint.
- o Enhancing ability to traverse and function with middleboxes.
- o A desire to use the port number space to enable load-sharing.

1.3.2. Reducing forwarding cost

It is a common requirement to terminate a large number of tunnels on a single router/host. The processing cost per tunnel includes both state (memory requirements) and per-packet processing at the tunnel ingress and egress.

Automatic IP Multicast Tunneling, known as AMT [I-D.ietf-mboned-auto-multicast] currently specifies UDP as the transport protocol for packets carrying tunneled IP multicast packets. The current specification for AMT states that the UDP checksum in the outer packet header should be zero (see Section 6.6 of [I-D.ietf-mboned-auto-multicast]). This argues that the computation of an additional checksum is an unwarranted burden on nodes implementing lightweight tunneling protocols when an inner packet is already adequately protected, . The AMT protocol needs to

replicate a multicast packet to each gateway tunnel. In this case, the outer IP addresses are different for each tunnel and therefore require a different pseudo header to be built for each UDP replicated encapsulation.

The argument concerning redundant processing costs is valid regarding the integrity of a tunneled packet. In some architectures (e.g. PC-based routers), other mechanisms may also significantly reduce checksum processing costs: There are implementations that have optimised checksum processing algorithms, including the use of checksum-offloading. This processing is readily available for IPv4 packets at high line rates. Such processing may be anticipated for IPv6 endpoints, allowing receivers to reject corrupted packets without further processing. However, there are certain classes of tunnel end-points where this off-loading is not available and unlikely to become available in the near future.

1.3.3. Need to inspect the entire packet

The currently-deployed hardware in many routers uses a fast-path processing that only provides the first n bytes of a packet to the forwarding engine, where typically $n \leq 128$.

When this design is used to support a tunnel ingress and egress, it prevents fast processing of a transport checksum over an entire (large) packet. Hence the currently defined IPv6 UDP checksum is poorly suited to use within a router that is unable to access the entire packet and does not provide checksum-offloading. Thus enabling checksum calculation over the complete packet can impact router design, performance improvement, energy consumption and/or cost.

1.3.4. Interactions with middleboxes

Many paths in the Internet include one or more middleboxes of various types. There exist large classes of middleboxes that will handle zero UDP checksum packets, which would not support UDP-Lite or the other investigated proposals. These middleboxes includes load balancers (see Section 1.3.5) including Equal Cost Multipath Routing, traffic classifiers and other functions that reads some fields in the UDP headers but does not validate the UDP checksum.

There are also middleboxes that either validates or modify the UDP checksum. The two most common classes are Firewalls and NATs. In IPv4, UDP-encapsulation may be desirable for NAT traversal, since UDP support is commonly provided. It is also necessary due to the almost ubiquitous deployment of IPv4 NATs. There has also been discussion of NAT for IPv6, although not for the same reason as in IPv4. If

IPv6 NAT becomes a reality they hopefully do not present the same protocol issues as for IPv4. If NAT is defined for IPv6, it should take into consideration the use of a zero UDP checksum.

The requirements for IPv6 firewall traversal are likely to be similar to those for IPv4. In addition, it can be reasonably expected that a firewall conforming to RFC 2460 will not regard datagrams with a zero UDP checksum as valid. Use of a zero UDP checksum with IPv6 requires firewalls to be updated before the full utility of the change is available.

It can be expected that datagrams with zero UDP checksum will initially not have the same middlebox traversal characteristics as regular UDP (RFC 2460). However when implementations follow the requirements specified in this document, we expect the traversal capabilities to improve over time. We also note that deployment of IPv6-capable middleboxes is still in its initial phases. Thus, it might be that the number of non-updated boxes quickly become a very small percentage of the deployed middleboxes.

1.3.5. Support for load balancing

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. An alternate method would be to utilise the IPv6 Flow Label [RFC6437] as a basis for entropy for load balancing. This would have the desirable effect of releasing IPv6 load-balancing devices from the need to assume semantics for the use of the transport port field and also works for all type of transport protocols.

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 Equal-Cost Multi-Path routing usage, (ECMP) [RFC6438] was produced. Router vendors could be encouraged to start using the IPv6 Flow Label as a part of the flow hash, providing support for ECMP without requiring use of UDP.

However, the method for populating the outer IPv6 header with a value for the flow label is not trivial: If the inner packet uses IPv6, then the flow label value could be copied to the outer packet header. However, many current end-points set the flow label to a zero value (thus no entropy). The ingress of a tunnel seeking to provide good entropy in the flow label field would therefore need to create a random flow label value and keep corresponding state, so that all packets that were associated with a flow would be consistently given the same flow label. Although possible, this complexity may not be

desirable in a tunnel ingress.

The end-to-end use of flow labels for load balancing is a long-term solution. Even if the usage of the flow label is clarified, there would be a transition time before a significant proportion of end-points start to assign a good quality flow label to the flows that they originate, with continued use of load balancing using the transport header fields until any widespread deployment is finally achieved.

2. Standards-Track Transports

The IETF has defined a set of transport protocols that may be applicable for tunnels with IPv6. There are also a set of network layer encapsulation tunnels such as IP-in-IP and GRE. These already standardized solutions are discussed here prior to the issues, as background for the issue description and some comparison of where the issue may already occur.

2.1. UDP with Standard Checksum

UDP [RFC0768] with standard checksum behaviour, as defined in RFC 2460, has already been discussed. UDP usage guidelines are provided in [RFC5405].

2.2. UDP-Lite

UDP-Lite [RFC3828] offers an alternate transport to UDP, specified as a proposed standard, RFC 3828. A MIB is defined in [RFC5097] and unicast usage guidelines in [RFC5405]. There is at least one open source implementation as a part of the Linux kernel since version 2.6.20.

UDP-Lite provides a checksum with optional partial coverage. When using this option, a datagram is divided into a sensitive part (covered by the checksum) and an insensitive part (not covered by the checksum). When the checksum covers the entire packet, UDP-Lite is fully equivalent with UDP, with the exception that it uses a different value in the Next Header field in the IPv6 header. Errors/corruption in the insensitive part will not cause the datagram to be discarded by the transport layer at the receiving endpoint. A minor side-effect of using UDP-Lite is that this was specified for damage-tolerant payloads and some link-layers may employ different link encapsulations when forwarding UDP-Lite segments (e.g. radio access bearers). Most link-layers will cover the insensitive part with the same strong layer 2 frame CRC that covers the sensitive part.

2.2.1. Using UDP-Lite as a Tunnel Encapsulation

Tunnel encapsulations can use UDP-Lite (e.g. Control And Provisioning of Wireless Access Points, CAPWAP [RFC5415]), since UDP-Lite provides a transport-layer checksum, including an IP pseudo header checksum, in IPv6, without the need for a router/middlebox to traverse the entire packet payload. This provides most of the verification required for delivery and still keeps a low complexity for the checksumming operation. UDP-Lite may set the length of checksum coverage on a per packet basis. This feature could be used if a tunnel protocol is designed to only verify delivery of the tunneled payload and uses a calculated checksum for control information.

There is currently poor support for middlebox traversal using UDP-Lite, because UDP-Lite uses a different IPv6 network-layer Next Header value to that of UDP, and few middleboxes are able to interpret UDP-Lite and take appropriate actions when forwarding the packet. This makes UDP-Lite less suited to protocols needing general Internet support, until such time that UDP-Lite has achieved better support in middleboxes and end-points.

2.3. General Tunnel Encapsulations

The IETF has defined a set of tunneling protocols or network layer encapsulations, e.g., IP-in-IP and GRE. These either do not include a checksum or use a checksum that is optional, since tunnel encapsulations are typically layered directly over the Internet layer (identified by the upper layer type in the IPv6 Next Header field) and are also not used as endpoint transport protocols. There is little chance of confusing a tunnel-encapsulated packet with other application data that could result in corruption of application state or data.

From the end-to-end perspective, the principal difference is that the network-layer Next Header field identifies a separate transport, which reduces the probability that corruption could result in the packet being delivered to the wrong endpoint or application. Specifically, packets are only delivered to protocol modules that process a specific Next Header value. The Next Header field therefore provides a first-level check of correct demultiplexing. In contrast, the UDP port space is shared by many diverse applications and therefore UDP demultiplexing relies solely on the port numbers.

2.4. Relation to UDP-Lite and UDP with checksum

The operation of IPv6 with UDP with a zero-checksum is not the same as IPv4 with UDP with a zero-checksum. Protocol designers should not

be fooled into thinking the two are the same. The requirements below list a set of additional considerations.

Where possible, existing general tunnel encapsulations, such as GRE, IP-in-IP, should be used. This section assumes that such existing tunnel encapsulations do not offer the functionality required to satisfy the protocol designer's goals. The section considers the standardized alternative solutions, rather than the full set of ideas evaluated in Appendix A. The alternatives to UDP with a zero checksum are UDP with a (calculated) checksum, and UDP-Lite.

UDP with a checksum has the advantage of close to universal support in both endpoints and middleboxes. It also provides statistical verification of delivery to the intended destination (address and port). However, some classes of device have limited support for calculation of a checksum that covers a full datagram. For these devices, this can incur significant processing cost (e.g. requiring processing in the router slow-path) and can hence reduce capacity or fail to function.

UDP-Lite has the advantage of using a checksum that is calculated only over the pseudo header and the UDP header. This provides a statistical verification of delivery to the intended destination (address and port). The checksum can be calculated without access to the datagram payload, only requiring access to the part to be protected. A drawback is that UDP-Lite has currently limited support in both end-points (i.e. is not supported on all operating system platforms) and middleboxes (that require support for the UDP-Lite header type). A path verification method is therefore recommended.

IPv6 and UDP with a zero-checksum can also be used by nodes that do not permit calculation of a payload checksum. Many existing classes of middleboxes do not verify or change the transport checksum. For these middleboxes, IPv6 with a zero UDP checksum is expected to function where UDP-Lite would not. However, support for the zero UDP checksum in middleboxes that do change or verify the checksum is currently limited, and this may result in datagrams with a zero UDP checksum being discarded, therefore a path verification method is recommended.

There are sets of constraints for which no solution exist: A protocol designer that needs to originate or receive datagrams on a device that can not efficiently calculate a checksum over a full datagram and also needs these packets to pass through a middlebox that verifies or changes a UDP checksum, but does not support a zero UDP checksum, can not use the zero UDP checksum method. Similarly, one that originates datagrams on a device with UDP-Lite support, but needs the packets to pass through a middlebox that does not support

UDP-Lite, can not use UDP-Lite. For such cases, there is no optimal solution and the current recommendation is to use or fall-back to using UDP with full checksum coverage.

3. Issues Requiring Consideration

This informative section evaluates issues around the proposal to update IPv6 [RFC2460], to enable the UDP transport checksum to be set to zero. Some of the identified issues are shared with other protocols already in use. The section also provides background to the requirements and recommendations that follow.

The decision in RFC 2460 to omit an integrity check at the network level meant that the IPv6 transport checksum was overloaded with many functions, including validating:

- o the endpoint address was not corrupted within a router, i.e., a packet was intended to be received by this destination and validate that the packet does not consist of a wrong header spliced to a different payload;
- o that extension header processing is correctly delimited - i.e., the start of data has not been corrupted. In this case, reception of a valid Next Header value provides some protection;
- o reassembly processing, when used;
- o the length of the payload;
- o the port values - i.e., the correct application receives the payload (applications should also check the expected use of source ports/addresses);
- o the payload integrity.

In IPv4, the first four checks are performed using the IPv4 header checksum.

In IPv6, these checks occur within the endpoint stack using the UDP checksum information. An IPv6 node also relies on the header information to determine whether to send an ICMPv6 error message [RFC4443] and to determine the node to which this is sent. Corrupted information may lead to misdelivery to an unintended application socket on an unexpected host.

3.1. Effect of packet modification in the network

IP packets may be corrupted as they traverse an Internet path. Older evidence in "When the CRC and TCP Checksum Disagree" [Sigcomm2000] show that this was once an issue in year 2000 with IPv4 routers, and occasional corruption could result from bad internal router processing in routers or hosts. These errors are not detected by the strong frame checksums employed at the link-layer [RFC3819]. During the development of this document in 2009, individuals provided reports of observed rates for received UDP datagrams using IPv4 where the UDP checksum had been detected as corrupt. These rates were as high as 1.39E-4 for some paths, but also close to zero for some other paths.

There is extensive experience of deployment using tunnel protocols in well-managed networks (e.g. corporate networks or service provider core networks). This has shown the robustness of methods such as PWE and MPLS that do not employ a transport protocol checksum and have not specified mechanisms to protect from corruption of the unprotected headers (such as the VPN Identifier in MPLS). Reasons for the robustness may include:

- o A reduced probability of corruption on paths through well-managed networks.
- o IP forms the majority of the inner traffic carried by these tunnels. Hence from a transport perspective, endpoint verification is already being performed when processing a received IPv4 packet or by the transport pseudo-header for an IPv6 packet. This update to UDP does not change this behaviour.
- o In certain cases, a combination of additional filtering (e.g. filter of a MAC destination address in a L2 tunnel) significantly reduces the probability of final mis-delivery to the IP stack.
- o The tunnel protocols did not use a UDP transport header, any corruption is therefore unlikely to result in misdelivery to another UDP-based application. This concern is specific to the use of UDP with IPv6.

While this experience can guide the present recommendations, any update to UDP must preserve operation in the general Internet. This is heterogeneous and can include links and systems of very varying characteristics. Transport protocols used by hosts need to be designed with this in mind, especially when there is need to traverse edge networks, where middlebox deployments are common.

For the general Internet, there is no current evidence that

corruption is rare, nor that this may not be applicable to IPv6. It therefore seems prudent not to relax checks on misdelivery. The emergence of low-end IPv6 routers and the proposed use of NAT with IPv6 further motivate the need to protect from misdelivery.

Corruption in the network may result in:

- o A datagram being misdelivered to the wrong host/router or the wrong transport entity within an endpoint. Such a datagram needs to be discarded;
- o A datagram payload being corrupted, but still delivered to the intended host/router transport entity. Such a datagram needs to be either discarded or correctly processed by an application that provides its own integrity checks;
- o A datagram payload being truncated by corruption of the length field. Such a datagram needs to be discarded.

When a checksum is used, this significantly reduces the impact of errors, reducing the probability of undetected corruption of state (and data) on both the host stack and the applications using the transport service.

The following sections examine the impact of modifying each of these header fields.

3.1.1. Corruption of the destination IP address

An IPv6 endpoint destination address could be modified in the network (e.g. corrupted by an error). This is not a concern for IPv4, because the IP header checksum will result in this packet being discarded by the receiving IP stack. Such modification in the network can not be detected at the network layer when using IPv6. Detection of this corruption by a UDP receiver relies on the IPv6 pseudo header incorporated in the transport checksum.

There are two possible outcomes:

- o Delivery to a destination address that is not in use (the packet will not be delivered, but could result in an error report);
- o Delivery to a different destination address. This modification will normally be detected by the transport checksum, resulting in silent discard. Without a computed checksum, the packet would be passed to the endpoint port demultiplexing function. If an application is bound to the associated ports, the packet payload will be passed to the application (see the subsequent section on

port processing).

3.1.2. Corruption of the source IP address

This section examines what happens when the source address is corrupted in transit. This is not a concern in IPv4, because the IP header checksum will normally result in this packet being discarded by the receiving IP stack. Detection of this corruption by a UDP receiver relies on the IPv6 pseudo header incorporated in the transport checksum.

Corruption of an IPv6 source address does not result in the IP packet being delivered to a different endpoint protocol or destination address. If only the source address is corrupted, the datagram will likely be processed in the intended context, although with erroneous origin information. When using Unicast Reverse Path Forwarding [RFC2827], a change in address may result in the router discarding the packet when the route to the modified source address is different to that of the source address of the original packet.

The result will depend on the application or protocol that processes the packet. Some examples are:

- o An application that requires a per-established context may disregard the datagram as invalid, or could map this to another context (if a context for the modified source address was already activated).
- o A stateless application will process the datagram outside of any context, a simple example is the ECHO server, which will respond with a datagram directed to the modified source address. This would create unwanted additional processing load, and generate traffic to the modified endpoint address.
- o Some datagram applications build state using the information from packet headers. A previously unused source address would result in receiver processing and the creation of unnecessary transport-layer state at the receiver. For example, Real Time Protocol (RTP) [RFC3550] sessions commonly employ a source independent receiver port. State is created for each received flow. Reception of a datagram with a corrupted source address will therefore result in accumulation of unnecessary state in the RTP state machine, including collision detection and response (since the same synchronization source, SSRC, value will appear to arrive from multiple source IP addresses).
- o ICMP messages relating to a corrupted packet can be misdirected to the wrong source node.

In general, the effect of corrupting the source address will depend upon the protocol that processes the packet and its robustness to this error. For the case where the packet is received by a tunnel endpoint, the tunnel application is expected to correctly handle a corrupted source address.

The impact of source address modification is more difficult to quantify when the receiving application is not that originally intended and several fields have been modified in transit.

3.1.3. Corruption of Port Information

This section describes what happens if one or both of the UDP port values are corrupted in transit. This can also happen with IPv4 is used with a zero UDP checksum, but not when UDP checksums are calculated or when UDP-Lite is used. If the ports carried in the transport header of an IPv6 packet were corrupted in transit, packets may be delivered to the wrong application process (on the intended machine) and/or responses or errors sent to the wrong application process (on the intended machine).

3.1.4. Delivery to an unexpected port

If one combines the corruption effects, such as destination address and ports, there is a number of potential outcomes when traffic arrives at an unexpected port. This section discusses these possibilities and their outcomes for a packet that does not use the UDP checksum validation:

- o Delivery to a port that is not in use. The packet is discarded, but could generate an ICMPv6 message (e.g. port unreachable).
- o It could be delivered to a different node that implements the same application, where the packet may be accepted, generating side-effects or accumulated state.
- o It could be delivered to an application that does not implement the tunnel protocol, where the packet may be incorrectly parsed, and may be misinterpreted, generating side-effects or accumulated state.

The probability of each outcome depends on the statistical probability that the address or the port information for the source or destination becomes corrupt in the datagram such that they match those of an existing flow or server port. Unfortunately, such a match may be more likely for UDP than for connection-oriented transports, because:

1. There is no handshake prior to communication and no sequence numbers (as in TCP, DCCP, or SCTP). Together, this makes it hard to verify that an application process is given only the application data associated with a specific transport session.
2. Applications writers often bind to wild-card values in endpoint identifiers and do not always validate correctness of datagrams they receive (guidance on this topic is provided in [RFC5405]).

While these rules could, in principle, be revised to declare naive applications as "Historic". This remedy is not realistic: the transport owes it to the stack to do its best to reject bogus datagrams.

If checksum coverage is suppressed, the application therefore needs to provide a method to detect and discard the unwanted data. A tunnel protocol would need to perform its own integrity checks on any control information if transported in datagrams with a zero UDP checksum. If the tunnel payload is another IP packet, the packets requiring checksums can be assumed to have their own checksums provided that the rate of corrupted packets is not significantly larger due to the tunnel encapsulation. If a tunnel transports other inner payloads that do not use IP, the assumptions of corruption detection for that particular protocol must be fulfilled, this may require an additional checksum/CRC and/or integrity protection of the payload and tunnel headers.

A protocol that uses a zero UDP checksum can not assume that it is the only protocol using a zero UDP checksum. Therefore, it needs to gracefully handle misdelivery. It must be robust to reception of malformed packets received on a listening port and expect that these packets may contain corrupted data or data associated with a completely different protocol.

3.1.5. Corruption of Fragmentation Information

The fragmentation information in IPv6 employs a 32-bit identity field, compared to only a 16-bit field in IPv4, a 13-bit fragment offset and a 1-bit flag, indicating if there are more fragments. Corruption of any of these field may result in one of two outcomes:

Reassembly failure: An error in the "More Fragments" field for the last fragment will for example result in the packet never being considered complete and will eventually be timed out and discarded. A corruption in the ID field will result in the fragment not being delivered to the intended context thus leaving the rest incomplete, unless that packet has been duplicated prior to corruption. The incomplete packet will eventually be timed out

and discarded.

Erroneous reassembly: The re-assembled packet did not match the original packet. This can occur when the ID field of a fragment is corrupted, resulting in a fragment becoming associated with another packet and taking the place of another fragment. Corruption in the offset information can cause the fragment to be misaligned in the reassembly buffer, resulting in incorrect reassembly. Corruption can cause the packet to become shorter or longer, however completion of reassembly is much less probable, since this would require consistent corruption of the IPv6 headers payload length field and the offset field. The possibility of mis-assembly requires the reassembling stack to provide strong checks that detect overlap or missing data, note however that this is not guaranteed and has been clarified in "Handling of Overlapping IPv6 Fragments" [RFC5722].

The erroneous reassembly of packets is a general concern and such packets should be discarded instead of being passed to higher layer processes. The primary detector of packet length changes is the IP payload length field, with a secondary check by the transport checksum. The Upper-Layer Packet length field included in the pseudo header assists in verifying correct reassembly, since the Internet checksum has a low probability of detecting insertion of data or overlap errors (due to misplacement of data). The checksum is also incapable of detecting insertion or removal of all zero-data that occurs in a multiple of a 16-bit chunk.

The most significant risk of corruption results following mis-association of a fragment with a different packet. This risk can be significant, since the size of fragments is often the same (e.g. fragments resulting when the path MTU results in fragmentation of a larger packet, common when addition of a tunnel encapsulation header expands the size of a packet). Detection of this type of error requires a checksum or other integrity check of the headers and the payload. Such protection is anyway desirable for tunnel encapsulations using IPv4, since the small fragmentation ID can easily result in wrap-around [RFC4963], this is especially the case for tunnels that perform flow aggregation [I-D.ietf-intarea-tunnels].

Tunnel fragmentation behavior matters. There can be outer or inner fragmentation "Tunnels in the Internet Architecture" [I-D.ietf-intarea-tunnels]. If there is inner fragmentation by the tunnel, the outer headers will never be fragmented and thus a zero UDP checksum in the outer header will not affect the reassembly process. When a tunnel performs outer header fragmentation, the tunnel egress needs to perform reassembly of the outer fragments into an inner packet. The inner packet is either a complete packet or a

fragment. If it is a fragment, the destination endpoint of the fragment will perform reassembly of the received fragments. The complete packet or the reassembled fragments will then be processed according to the packet Next Header field. The receiver may only detect reassembly anomalies when it uses a protocol with a checksum. The larger the number of reassembly processes to which a packet has been subjected, the greater the probability of an error.

- o An IP-in-IP tunnel that performs inner fragmentation has similar properties to a UDP tunnel with a zero UDP checksum that also performs inner fragmentation.
- o An IP-in-IP tunnel that performs outer fragmentation has similar properties to a UDP tunnel with a zero UDP checksum that performs outer fragmentation.
- o A tunnel that performs outer fragmentation can result in a higher level of corruption due to both inner and outer fragmentation, enabling more chances for reassembly errors to occur.
- o Recursive tunneling can result in fragmentation at more than one header level, even for inner fragmentation unless it goes to the inner-most IP header.
- o Unless there is verification at each reassembly, the probability for undetected error will increase with the number of times fragmentation is recursively applied, making IP-in-IP and UDP with zero UDP checksum both vulnerable to undetected errors.

In conclusion, fragmentation of datagrams with a zero UDP checksum does not worsen the performance compared to some other commonly used tunnel encapsulations. However, caution is needed for recursive tunneling without any additional verification at the different tunnel layers.

3.2. Where Packet Corruption Occurs

Corruption of IP packets can occur at any point along a network path, during packet generation, during transmission over the link, in the process of routing and switching, etc. Some transmission steps include a checksum or Cyclic Redundancy Check (CRC) that reduces the probability for corrupted packets being forwarded, but there still exists a probability that errors may propagate undetected.

Unfortunately the community lacks reliable information to identify the most common functions or equipment that result in packet corruption. However, there are indications that the place where corruption occurs can vary significantly from one path to another.

There is therefore a risk in applying evidence from one domain of usage to infer characteristics for another. Methods intended for general Internet usage must therefore assume that corruption can occur and deploy mechanisms to mitigate the effect of corruption and/or resulting misdelivery.

3.3. Validating the network path

IP transports designed for use in the general Internet should not assume specific path characteristics. Network protocols may reroute packets that change the set of routers and middleboxes along a path. Therefore transports such as TCP, SCTP and DCCP have been designed to negotiate protocol parameters, adapt to different network path characteristics, and receive feedback to verify that the current path is suited to the intended application. Applications using UDP and UDP-Lite need to provide their own mechanisms to confirm the validity of the current network path.

A zero value in the UDP checksum field is explicitly disallowed in RFC2460. Thus it may be expected that any device on the path that has a reason to look beyond the IP header, for example to validate the UDP checksum, will consider such a packet as erroneous or illegal and may discard it, unless the device is updated to support the new behavior. Any middlebox that modifies the UDP checksum, for example a NAT that changes the values of the IP and UDP header in such a way that the checksum over the pseudo header changes value, will need to be updated to support this behavior. Until then, a zero UDP checksum packet is likely to be discarded either directly in the middlebox or at the destination, when a zero UDP checksum has been modified to a non-zero by an incremental update.

A pair of end-points intending to use a new behavior will therefore not only need to ensure support at each end-point, but also that the path between them will deliver packets with the new behavior. This may require using negotiation or an explicit mandate to use the new behavior by all nodes that support the new protocol.

Enabling the use of a zero checksum places new requirements on equipment deployed within the network, such as middleboxes. A middlebox (e.g. Firewalls, Network Address Translators) may enable zero checksum usage for a particular range of ports. Note that checksum off-loading and operating system design may result in all IPv6 UDP traffic being sent with a calculated checksum. This requires middleboxes that are configured to enable a zero UDP checksum to continue to work with bidirectional UDP flows that use a zero UDP checksum in only one direction, and therefore they must not maintain separate state for a UDP flow based on its checksum usage.

Support along the path between end points can be guaranteed in limited deployments by appropriate configuration. In general, it can be expected to take time for deployment of any updated behaviour to become ubiquitous.

A sender will need to probe the path to verify the expected behavior. Path characteristics may change, and usage therefore should be robust and able to detect a failure of the path under normal usage and re-negotiate. Note that a bidirectional path does not necessarily support the same checksum usage in both the forward and return directions: Receipt of a datagram with a zero UDP checksum, does not imply that the remote endpoint can also receive a datagram with a zero UDP checksum. This will require periodic validation of the path, adding complexity to any solution using the new behavior.

3.4. Applicability of method

The update to the IPv6 specification defined in [I-D.ietf-6man-udpchecksums] only modifies IPv6 nodes that implement specific protocols designed to permit omission of a UDP checksum. This document therefore provides an applicability statement for the updated method indicating when the mechanism can (and can not) be used. Enabling this, and ensuring correct interactions with the stack, implies much more than simply disabling the checksum algorithm for specific packets at the transport interface.

When the method is widely available, it may be expected to be used by applications that are perceived to gain benefit. Any solution that uses an end-to-end transport protocol, rather than an IP-in-IP encapsulation, needs to minimise the possibility that application processes could confuse a corrupted or wrongly delivered UDP datagram with that of data addressed to the application running on their endpoint.

The protocol or application that uses the zero checksum method must ensure that the lack of checksum does not affect the protocol operation. This includes being robust to receiving a unintended packet from another protocol or context following corruption of a destination or source address and/or port value. It also includes considering the need for additional implicit protection mechanisms required when using the payload of a UDP packet received with a zero checksum.

3.5. Impact on non-supporting devices or applications

It is important to consider the potential impact of using a zero UDP checksum on end-point devices or applications that are not modified to support the new behavior or by default or preference, use the

regular behavior. These applications must not be significantly impacted by the update.

To illustrate why this necessary, consider the implications of a node that enables use of a zero UDP checksum at the interface level: This would result in all applications that listen to a UDP socket receiving datagrams where the checksum was not verified. This could have a significant impact on an application that was not designed with the additional robustness needed to handle received packets with corruption, creating state or destroying existing state in the application.

A zero UDP checksum therefore needs to be enabled only for individual ports using an explicit request by the application. In this case, applications using other ports would maintain the current IPv6 behavior, discarding incoming datagrams with a zero UDP checksum. These other applications would not be affected by this changed behavior. An application that allows the changed behavior should be aware of the risk of corruption and the increased level of misdirected traffic, and can be designed robustly to handle this risk.

4. Constraints on implementation of IPv6 nodes supporting zero checksum

This section is an applicability statement that defines requirements and recommendations on the implementation of IPv6 nodes that support use of a zero value in the checksum field of a UDP datagram.

All implementations that support this zero UDP checksum method **MUST** conform to the requirements defined below.

1. An IPv6 sending node **MAY** use a calculated RFC 2460 checksum for all datagrams that it sends. This explicitly permits an interface that supports checksum offloading to insert an updated UDP checksum value in all UDP datagrams that it forwards, however note that sending a calculated checksum requires the receiver to also perform the checksum calculation. Checksum offloading can normally be switched off for a particular interface to ensure that datagrams are sent with a zero UDP checksum.
2. IPv6 nodes **SHOULD** by default **NOT** allow the zero UDP checksum method for transmission.
3. IPv6 nodes **MUST** provide a way for the application/protocol to indicate the set of ports that will be enabled to send datagrams with a zero UDP checksum. This may be implemented by enabling a

transport mode using a socket API call when the socket is established, or a similar mechanism. It may also be implemented by enabling the method for a pre-assigned static port used by a specific tunnel protocol.

4. IPv6 nodes MUST provide a method to allow an application/protocol to indicate that a particular UDP datagram is required to be sent with a UDP checksum. This needs to be allowed by the operating system at any time (e.g. to send keep-alive datagrams), not just when a socket is established in the zero checksum mode.
5. The default IPv6 node receiver behaviour MUST discard all IPv6 packets carrying datagrams with a zero UDP checksum.
6. IPv6 nodes MUST provide a way for the application/protocol to indicate the set of ports that will be enabled to receive datagrams with a zero UDP checksum. This may be implemented via a socket API call, or similar mechanism. It may also be implemented by enabling the method for a pre-assigned static port used by a specific tunnel protocol.
7. IPv6 nodes supporting usage of zero UDP checksums MUST also allow reception using a calculated UDP checksum on all ports configured to allow zero UDP checksum usage. (The sending endpoint, e.g. encapsulating ingress, may choose to compute the UDP checksum, or may calculate this by default.) The receiving endpoint MUST use the reception method specified in RFC2460 when the checksum field is not zero.
8. RFC 2460 specifies that IPv6 nodes SHOULD log received datagrams with a zero UDP checksum. This remains the case for any datagram received on a port that does not explicitly enable processing of a zero UDP checksum. A port for which the zero UDP checksum has been enabled MUST NOT log the datagram solely because the checksum value is zero.
9. IPv6 nodes MAY separately identify received UDP datagrams that are discarded with a zero UDP checksum. It SHOULD NOT add these to the standard log, since the endpoint has not been verified. This may be used to support other functions (such as a security policy).
10. IPv6 nodes that receive ICMPv6 messages that refer to packets with a zero UDP checksum MUST provide appropriate checks concerning the consistency of the reported packet to verify that the reported packet actually originated from the node, before acting upon the information (e.g. validating the address and

port numbers in the ICMPv6 message body).

5. Requirements on usage of the zero UDP checksum

This section is an applicability statement that identifies requirements and recommendations for protocols and tunnel encapsulations that are transported over an IPv6 transport flow (e.g. tunnel) that does not perform a UDP checksum calculation to verify the integrity at the transport endpoints. Before deciding to use the zero UDP checksum and loose the integrity verification provided, a protocol developer should seriously consider if they can use checksummed UDP packets or UDP-Lite [RFC3828], because IPv6 with a zero UDP checksum is not equivalent in behavior to IPv4 with zero UDP checksum.

The requirements and recommendations for protocols and tunnel encapsulations using an IPv6 transport flow that does not perform a UDP checksum calculation to verify the integrity at the transport endpoints are:

1. Transported protocols that enable the use of zero UDP checksum MUST only enable this for a specific port or port-range. This needs to be enabled at the sending and receiving endpoints for a UDP flow.
2. An integrity mechanism is always RECOMMENDED at the transported protocol layer to ensure that corruption rates of the delivered payload is not increased (e.g. the inner-most packet of a UDP tunnel). A mechanism that isolates the causes of corruption (e.g. identifying misdelivery, IPv6 header corruption, tunnel header corruption) is expected to also provide additional information about the status of the tunnel (e.g. to suggest a security attack).
3. A transported protocol that encapsulates Internet Protocol (IPv4 or IPv6) packets MAY rely on the inner packet integrity checks, provided that the tunnel protocol will not significantly increase the rate of corruption of the inner IP packet. If a significantly increased corruption rate can occur, then the tunnel protocol MUST provide an additional integrity verification mechanism. Early detection is desirable to avoid wasting unnecessary computation, transmission capacity or storage for packets that will subsequently be discarded.
4. A transported protocol that supports use of a zero UDP checksum, MUST be designed so that corruption of this information does not result in accumulated state for the protocol.

5. A transported protocol with a non-tunnel payload or one that encapsulates non-IP packets **MUST** have a CRC or other mechanism for checking packet integrity, unless the non-IP packet is specifically designed for transmission over a lower layer that does not provide a packet integrity guarantee.
6. A transported protocol with control feedback **SHOULD** be robust to changes in the network path, since the set of middleboxes on a path may vary during the life of an association. The UDP endpoints need to discover paths with middleboxes that drop packets with a zero UDP checksum. Therefore, transported protocols **SHOULD** send keep-alive messages with a zero UDP checksum. An endpoint that discovers an appreciable loss rate for keep-alive packets **MAY** terminate the UDP flow (e.g. tunnel). Section 3.1.3 of RFC 5405 describes requirements for congestion control when using a UDP-based transport.
7. A protocol with control feedback that can fall-back to using UDP with a calculated RFC 2460 checksum is expected to be more robust to changes in the network path. Therefore, keep-alive messages **SHOULD** include both UDP datagrams with a checksum and datagrams with a zero UDP checksum. This will enable the remote endpoint to distinguish between a path failure and dropping of datagrams with a zero UDP checksum.
8. A middlebox implementation **MUST** allow forwarding of an IPv6 UDP datagram with both a zero and standard UDP checksum using the same UDP port.
9. A middlebox **MAY** configure a restricted set of specific port ranges that forward UDP datagrams with a zero UDP checksum. The middlebox **MAY** drop IPv6 datagrams with a zero UDP checksum that are outside a configured range.
10. When a middlebox forwards an IPv6 UDP flow containing datagrams with both a zero and standard UDP checksum, the middlebox **MUST** NOT maintain separate state for flows depending on the value of their UDP checksum field. (This requirement is necessary to enable a sender that always calculates a checksum to communicate via a middlebox with a remote endpoint that uses a zero UDP checksum.)

Special considerations are required when designing a UDP tunnel protocol, where the tunnel ingress or egress may be a router that may not have access to the packet payload. When the node is acting as a host (i.e., sending or receiving a packet addressed to itself), the checksum processing is similar to other hosts. However, when the node (e.g. a router) is acting as a tunnel ingress or egress that

forwards a packet to or from a UDP tunnel, there may be restricted access to the packet payload. This prevents calculating (or verifying) a UDP checksum. In this case, the tunnel protocol may use a zero UDP checksum and must:

- o Ensure that tunnel ingress and tunnel egress router are both configured to use a zero UDP checksum. For example, this may include ensuring that hardware checksum offloading is disabled.
- o The tunnel operator must ensure that middleboxes on the network path are updated to support use of a zero UDP checksum.
- o A tunnel egress should implement appropriate security techniques to protect from overload, including source address filtering to prevent traffic injection by an attacker, and rate-limiting of any packets that incur additional processing, such as UDP datagrams used for control functions that require verification of a calculated checksum to verify the network path. Usage of common control traffic for multiple tunnels between a pair of nodes can assist in reducing the number of packets to be processed.

6. Summary

This document provides an applicability statement for the use of UDP transport checksums with IPv6.

It examines the role of the UDP transport checksum when used with IPv6 and presents a summary of the trade-offs in evaluating the safety of updating RFC 2460 to permit an IPv6 endpoint to use a zero UDP checksum field to indicate that no checksum is present.

Application designers should first examine whether their transport goals may be met using standard UDP (with a calculated checksum) or by using UDP-Lite. The use of UDP with a zero UDP checksum has merits for some applications, such as tunnel encapsulation, and is widely used in IPv4. However, there are different dangers for IPv6: There is an increased risk of corruption and misdelivery when using zero UDP checksum in IPv6 compared to using IPv4 due to the lack of an IPv6 header checksum. Thus, applications need to evaluate the risks of enabling use of a zero UDP checksum and consider a solution that at least provides the same delivery protection as for IPv4, for example by utilizing UDP-Lite, or by enabling the UDP checksum. The use of checksum off-loading may help alleviate the cost of checksum processing and permit use of a checksum using method defined in RFC 2460.

Tunnel applications using UDP for encapsulation can in many cases use

a zero UDP checksum without significant impact on the corruption rate. A well-designed tunnel application should include consistency checks to validate the header information encapsulated with a received packet. In most cases, tunnels encapsulating IP packets can rely on the integrity protection provided by the transported protocol (or tunneled inner packet). When correctly implemented, such an endpoint will not be negatively impacted by omission of the transport-layer checksum. Recursive tunneling and fragmentation is a potential issue that can raise corruption rates significantly, and requires careful consideration.

Other UDP applications at the intended destination node or another node can be impacted if they are allowed to receive datagrams that have a zero UDP checksum. It is important that already deployed applications are not impacted by a change at the transport layer. If these applications execute on nodes that implement RFC 2460, they will discard (and log) all datagrams with a zero UDP checksum. This is not an issue.

In general, UDP-based applications need to employ a mechanism that allows a large percentage of the corrupted packets to be removed before they reach an application, both to protect the data stream of the application and the control plane of higher layer protocols. These checks are currently performed by the UDP checksum for IPv6, or the reduced checksum for UDP-Lite when used with IPv6.

The transport of recursive tunneling and the use of fragmentation pose difficult issues that need to be considered in the design of tunnel protocols. There is an increased risk of an error in the inner-most packet when fragmentation when several layers of tunneling and several different reassembly processes are run without verification of correctness. This requires extra thought and careful consideration in the design of transported tunnels.

Any use of the updated method must consider the implications on firewalls, NATs and other middleboxes. It is not expected that IPv6 NATs handle IPv6 UDP datagrams in the same way that they handle IPv4 UDP datagrams. In many deployed cases this will require an update to support an IPv6 zero UDP checksum. Firewalls are intended to be configured, and therefore may need to be explicitly updated to allow new services or protocols. IPv6 middlebox deployment is not yet as prolific as it is in IPv4, and therefore new devices are expected to follow the methods specified in this document.

Each application should consider the implications of choosing an IPv6 transport that uses a zero UDP checksum, and consider whether other standard methods may be more appropriate, and may simplify application design.

7. Acknowledgements

Brian Haberman, Brian Carpenter, Margaret Wasserman, Lars Eggert, others in the TSV directorate. Barry Leiba, Ronald Bonica, Pete Resnick, and Stewart Bryant are thanked for resulting in a document with much greater applicability. Thanks to P.F. Chimento for careful review and editorial corrections.

Thanks also to: Remi Denis-Courmont, Pekka Savola, Glen Turner, and many others who contributed comments and ideas via the 6man, behave, lisp and mboned lists.

8. IANA Considerations

This document does not require any actions by IANA.

9. Security Considerations

Transport checksums provide the first stage of protection for the stack, although they can not be considered authentication mechanisms. These checks are also desirable to ensure packet counters correctly log actual activity, and can be used to detect unusual behaviours.

Depending on the hardware design, the processing requirements may differ for tunnels that have a zero UDP checksum and those that calculate a checksum. This processing overhead may need to be considered when deciding whether to enable a tunnel and to determine an acceptable rate for transmission. This can become a security risk for designs that can handle a significantly larger number of packets with zero UDP checksums compared to datagrams with a non-zero checksum, such as tunnel egress. An attacker could attempt to inject non-zero checksummed UDP packets into a tunnel forwarding zero checksum UDP packets and cause overload in the processing of the non-zero checksums, e.g. if this happens in a routers slow path. Protection mechanisms should therefore be employed when this threat exists. Protection may include source address filtering to prevent an attacker injecting traffic, as well as throttling the amount of non-zero checksum traffic. The latter may impact the function of the tunnel protocol.

Transmission of IPv6 packets with a zero UDP checksum could reveal additional information to an on-path attacker to identify the operating system or configuration of a sending node. There is a need to probe the network path to determine whether the current path supports using IPv6 packets with a zero UDP checksum. The details of the probing mechanism may differ for different tunnel encapsulations

and if visible in the network (e.g. if not using IPsec in encryption mode) could reveal additional information to an on-path attacker to identify the type of tunnel being used.

IP-in-IP or GRE tunnels offer good traversal of middleboxes that have not been designed for security, e.g. firewalls. However, firewalls may be expected to be configured to block general tunnels as they present a large attack surface. This applicability statement therefore permits this method to be enabled only for specific ranges of ports.

When the zero UDP checksum mode is enabled for a range of ports, nodes and middleboxes must forward received UDP datagrams that have either a calculated checksum or a zero checksum.

10. References

10.1. Normative References

- [I-D.ietf-6man-udpchecksums]
Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", draft-ietf-6man-udpchecksums-08 (work in progress), February 2013.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

10.2. Informative References

- [I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-00 (work in progress), March 2010.
- [I-D.ietf-mboned-auto-multicast]
Bumgardner, G., "Automatic Multicast Tunneling", draft-ietf-mboned-auto-multicast-14 (work in progress),

June 2012.

- [LISP] D. Farinacci et al, "Locator/ID Separation Protocol (LISP)", November 2012.
- [RFC1071] Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum", RFC 1071, September 1988.
- [RFC1141] Mallory, T. and A. Kullberg, "Incremental updating of the Internet checksum", RFC 1141, January 1990.
- [RFC1624] Rijssinghani, A., "Computation of the Internet Checksum via Incremental Update", RFC 1624, May 1994.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.
- [RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.
- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.
- [RFC5097] Renker, G. and G. Fairhurst, "MIB for the UDP-Lite protocol", RFC 5097, January 2008.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.
- [RFC5415] Calhoun, P., Montemurro, M., and D. Stanley, "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol

Specification", RFC 5415, March 2009.

[RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, December 2009.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, November 2011.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, November 2011.

[Sigcomm2000] Jonathan Stone and Craig Partridge , "When the CRC and TCP Checksum Disagree", 2000.

[UDPTT] G Fairhurst, "The UDP Tunnel Transport mode", Feb 2010.

Appendix A. Evaluation of proposal to update RFC 2460 to support zero checksum

This informative appendix documents the evaluation of the proposal to update IPv6 [RFC2460], to provide the option that some nodes may suppress generation and checking of the UDP transport checksum. It also compares the proposal with other alternatives, and notes that for a particular application some standard methods may be more appropriate than using IPv6 with a zero UDP checksum.

A.1. Alternatives to the Standard Checksum

There are several alternatives to the normal method for calculating the UDP Checksum [RFC1071] that do not require a tunnel endpoint to inspect the entire packet when computing a checksum. These include (in decreasing order of complexity):

- o Delta computation of the checksum from an encapsulated checksum field. Since the checksum is a cumulative sum [RFC1624], an encapsulating header checksum can be derived from the new pseudo header, the inner checksum and the sum of the other network-layer fields not included in the pseudo header of the encapsulated packet, in a manner resembling incremental checksum update [RFC1141]. This would not require access to the whole packet, but does require fields to be collected across the header, and arithmetic operations on each packet. The method would only work for packets that contain a 2's complement transport checksum (i.e., it would not be appropriate for SCTP or when IP fragmentation is used).

- o UDP-Lite with the checksum coverage set to only the header portion of a packet. This requires a pseudo header checksum calculation only on the encapsulating packet header. The computed checksum value may be cached (before adding the Length field) for each flow/destination and subsequently combined with the Length of each packet to minimise per-packet processing. This value is combined with the UDP payload length for the pseudo header, however this length is expected to be known when performing packet forwarding.
- o The proposed UDP Tunnel Transport [UDPTT] suggested a method where UDP would be modified to derive the checksum only from the encapsulating packet protocol header. This value does not change between packets in a single flow. The value may be cached per flow/destination to minimise per-packet processing.
- o There has been a proposal to simply ignore the UDP checksum value on reception at the tunnel egress, allowing a tunnel ingress to insert any value correct or false. For tunnel usage, a non standard checksum value may be used, forcing an RFC 2460 receiver to drop the packet. The main downside is that it would be impossible to identify a UDP datagram (in the network or an endpoint) that is treated in this way compared to a packet that has actually been corrupted.
- o A method has been proposed that uses a new (to be defined) IPv6 Destination Options Header to provide an end-to-end validation check at the network layer. This would allow an endpoint to verify delivery to an appropriate end point, but would also require IPv6 nodes to correctly handle the additional header, and would require changes to middlebox behavior (e.g. when used with a NAT that always adjusts the checksum value).
- o UDP modified to disable checksum processing [I-D.ietf-6man-udpchecksums]. This eliminates the need for a checksum calculation, but would require constraints on appropriate usage and updates to end-points and middleboxes.
- o IP-in-IP tunneling. As this method completely dispenses with a transport protocol in the outer-layer it has reduced overhead and complexity, but also reduced functionality. There is no outer checksum over the packet and also no ports to perform demultiplexing between different tunnel types. This reduces the information available upon which a load balancer may act.

These options are compared and discussed further in the following sections.

A.2. Comparison

This section compares the above listed methods to support datagram tunneling. It includes proposals for updating the behaviour of UDP.

While this comparison focuses on applications that are expected to execute on routers, the distinction between a router and a host is not always clear, especially at the transport level. Systems (such as unix-based operating systems) routinely provide both functions. There is no way to identify the role of the receiving node from a received packet.

A.2.1. Middlebox Traversal

Regular UDP with a standard checksum or the delta encoded optimization for creating correct checksums have the best possibilities for successful traversal of a middlebox. No new support is required.

A method that ignores the UDP checksum on reception is expected to have a good probability of traversal, because most middleboxes perform an incremental checksum update. UDPTT would also have been able to traverse a middlebox with this behaviour. However, a middlebox on the path that attempts to verify a standard checksum will not forward packets using either of these methods, preventing traversal. A method that ignores the checksum has an additional downside in that it prevents improvement of middlebox traversal, because there is no way to identify UDP datagrams that use the modified checksum behaviour.

IP-in-IP or GRE tunnels offer good traversal of middleboxes that have not been designed for security, e.g. firewalls. However, firewalls may be expected to be configured to block general tunnels as they present a large attack surface.

A new IPv6 Destination Options header will suffer traversal issues with middleboxes, especially Firewalls and NATs, and will likely require them to be updated before the extension header is passed.

Datagrams with a zero UDP checksum will not be passed by any middlebox that validates the checksum using RFC 2460 or updates the checksum field, such as NAT or firewalls. This would require an update to correctly handle a datagram with a zero UDP checksum.

UDP-Lite will require an update of almost all type of middleboxes, because it requires support for a separate network-layer protocol number. Once enabled, the method to support incremental checksum update would be identical to that for UDP, but different for checksum

validation.

A.2.2. Load Balancing

The usefulness of solutions for load balancers depends on the difference in entropy in the headers for different flows that can be included in a hash function. All the proposals that use the UDP protocol number have equal behavior. UDP-Lite has the potential for equally good behavior as for UDP. However, UDP-Lite is currently unlikely to be supported by deployed hashing mechanisms, which could cause a load balancer to not use the transport header in the computed hash. A load balancer that only uses the IP header will have low entropy, but could be improved by including the IPv6 the flow label, providing that the tunnel ingress ensures that different flow labels are assigned to different flows. However, a transition to the common use of good quality flow labels is likely to take time to deploy.

A.2.3. Ingress and Egress Performance Implications

IP-in-IP tunnels are often considered efficient, because they introduce very little processing and low data overhead. The other proposals introduce a UDP-like header incurring associated data overhead. Processing is minimised for the method that uses a zero UDP checksum, ignoring the UDP checksum on reception, and only slightly higher for UDPTT, the extension header and UDP-Lite. The delta-calculation scheme operates on a few more fields, but also introduces serious failure modes that can result in a need to calculate a checksum over the complete datagram. Regular UDP is clearly the most costly to process, always requiring checksum calculation over the entire datagram.

It is important to note that the zero UDP checksum method, ignoring checksum on reception, the Option Header, UDPTT and UDP-Lite will likely incur additional complexities in the application to incorporate a negotiation and validation mechanism.

A.2.4. Deployability

The major factors influencing deployability of these solutions are a need to update both end-points, a need for negotiation and the need to update middleboxes. These are summarised below:

- o The solution with the best deployability is regular UDP. This requires no changes and has good middlebox traversal characteristics.
- o The next easiest to deploy is the delta checksum solution. This does not modify the protocol on the wire and only needs changes in

tunnel ingress.

- o IP-in-IP tunnels should not require changes to the end-points, but raise issues when traversing firewalls and other security devices, which are expected to require updates.
- o Ignoring the checksum on reception will require changes at both end-points. The never ceasing risk of path failure requires additional checks to ensure this solution is robust and will require changes or additions to the tunnel control protocol to negotiate support and validate the path.
- o The remaining solutions (including the zero checksum method) offer similar deployability. UDP-Lite requires support at both end-points and in middleboxes. UDPTT and the zero UDP checksum method with or without an extension header require support at both end-points and in middleboxes. UDP-Lite, UDPTT, and the zero UDP checksum method and use of extension headers may additionally require changes or additions to the tunnel control protocol to negotiate support and path validation.

A.2.5. Corruption Detection Strength

The standard UDP checksum and the delta checksum can both provide some verification at the tunnel egress. This can significantly reduce the probability that a corrupted inner packet is forwarded. UDP-Lite, UDPTT and the extension header all provide some verification against corruption, but do not verify the inner packet. They only provide a strong indication that the delivered packet was intended for the tunnel egress and was correctly delimited.

The methods using a zero UDP checksum, ignoring the UDP checksum on reception and IP-and-IP encapsulation all provide no verification that a received datagram was intended to be processed by a specific tunnel egress or that the inner encapsulated packet was correct. Section 3.1 discusses experience using specific protocols in well-managed networks.

A.2.6. Comparison Summary

The comparisons above may be summarised as "there is no silver bullet that will slay all the issues". One has to select which down side(s) can best be lived with. Focusing on the existing solutions, this can be summarized as:

Regular UDP: The method defined in RFC 2460 has good middlebox traversal and load balancing and multiplexing, requiring a checksum in the outer headers covering the whole packet.

IP in IP: A low complexity encapsulation, with limited middlebox traversal, no multiplexing support, and currently poor load balancing support that could improve over time.

UDP-Lite: A medium complexity encapsulation, with good multiplexing support, limited middlebox traversal, but possible to improve over time, currently poor load balancing support that could improve over time, in most cases requiring application level negotiation to select the protocol and validation to confirm the path forwards UDP-Lite.

The delta-checksum is an optimization in the processing of UDP, as such it exhibits some of the drawbacks of using regular UDP.

The remaining proposals may be described in similar terms:

Zero-Checksum: A low complexity encapsulation, with good multiplexing support, limited middlebox traversal that could improve over time, good load balancing support, in most cases requiring application level negotiation and validation to confirm the path forwards a zero UDP checksum.

UDPTT: A medium complexity encapsulation, with good multiplexing support, limited middlebox traversal, but possible to improve over time, good load balancing support, in most cases requiring application level negotiation to select the transport and validation to confirm the path forwards UDPTT datagrams.

IPv6 Destination Option IP in IP tunneling: A medium complexity, with no multiplexing support, limited middlebox traversal, currently poor load balancing support that could improve over time, in most cases requiring negotiation to confirm the option is supported and validation to confirm the path forwards the option.

IPv6 Destination Option combined with UDP Zero-checksumming: A medium complexity encapsulation, with good multiplexing support, limited load balancing support that could improve over time, in most cases requiring negotiation to confirm the option is supported and validation to confirm the path forwards the option.

Ignore the checksum on reception: A low complexity encapsulation, with good multiplexing support, medium middlebox traversal that never can improve, good load balancing support, in most cases requiring negotiation to confirm the option is supported by the

remote endpoint and validation to confirm the path forwards a zero UDP checksum.

There is no clear single optimum solution. If the most important need is to traverse middleboxes, then the best choice is to stay with regular UDP and consider the optimizations that may be required to perform the checksumming. If one can live with limited middlebox traversal, low complexity is necessary and one does not require load balancing, then IP-in-IP tunneling is the simplest. If one wants strengthened error detection, but with currently limited middlebox traversal and load-balancing. UDP-Lite is appropriate. Zero UDP checksum addresses another set of constraints, low complexity and a need for load balancing from the current Internet, providing it can live with currently limited middlebox traversal.

Techniques for load balancing and middlebox traversal do continue to evolve. Over a long time, developments in load balancing have good potential to improve. This time horizon is long since it requires both load balancer and end-point updates to get full benefit. The challenges of middlebox traversal are also expected to change with time, as device capabilities evolve. Middleboxes are very prolific with a larger proportion of end-user ownership, and therefore may be expected to take long time cycles to evolve.

One potential advantage is that the deployment of IPv6-capable middleboxes are still in its initial phase and the quicker a new method becomes standardized, the fewer boxes will be non-compliant.

Thus, the question of whether to permit use of datagrams with a zero UDP checksum for IPv6 under reasonable constraints, is therefore best viewed as a trade-off between a number of more subjective questions:

- o Is there sufficient interest in using a zero UDP checksum with the given constraints (summarised below)?
- o Are there other avenues of change that will resolve the issue in a better way and sufficiently quickly ?
- o Do we accept the complexity cost of having one more solution in the future?

The analysis concludes that the IETF should carefully consider constraints on sanctioning the use of any new transport mode. The 6man working group of the IETF has determined that the answer to the above questions are sufficient to update IPv6 to standardise use of a zero UDP checksum for use by tunnel encapsulations for specific applications.

Each application should consider the implications of choosing an IPv6 transport that uses a zero UDP checksum. In many cases, standard methods may be more appropriate, and may simplify application design. The use of checksum off-loading may help alleviate the checksum processing cost and permit use of a checksum using method defined in RFC 2460.

Appendix B. Document Change History

{RFC EDITOR NOTE: This section must be deleted prior to publication}

Individual Draft 00 This is the first DRAFT of this document - It contains a compilation of various discussions and contributions from a variety of IETF WGs, including: mboned, tsv, 6man, lisp, and behave. This includes contributions from Magnus with text on RTP, and various updates.

Individual Draft 01

- * This version corrects some typos and editorial NiTs and adds discussion of the need to negotiate and verify operation of a new mechanism (3.3.4).

Individual Draft 02

- * Version -02 corrects some typos and editorial NiTs.
- * Added reference to ECMP for tunnels.
- * Clarifies the recommendations at the end of the document.

Working Group Draft 00

- * Working Group Version -00 corrects some typos and removes much of rationale for UDPTT. It also adds some discussion of IPv6 extension header.

Working Group Draft 01

- * Working Group Version -01 updates the rules and incorporates off-list feedback. This version is intended for wider review within the 6man working group.

Working Group Draft 02

- * This version is the result of a major rewrite and re-ordering of the document.
- * A new section comparing the results have been added.
- * The constraints list has been significantly altered by removing some and rewording other constraints.
- * This contains other significant language updates to clarify the intent of this draft.

Working Group Draft 03

- * Editorial updates

Working Group Draft 04

- * Resubmission only updating the AMT and RFC2765 references.

Working Group Draft 05

- * Resubmission to correct editorial NiTs - thanks to Bill Atwood for noting these. Group Draft 05.

Working Group Draft 06

- * Resubmission to keep draft alive (spelling updated from 05).

Working Group Draft 07

- * Interim Version
- * Submission after IESG Feedback Added
- * Updates to enable the document to become a PS Applicability Statement

Working Group Draft 08

- * First Version written as a PS Applicability Statement
- * Changes to reflect decision to update RFC 2460, rather than recommend decision
- * Updates to requirements for middleboxes

- * Inclusion of requirements for security, API, and tunnel
- * Move of the rationale for the update to an Annex (former section 4)

Working Group Draft 09

- * Submission after second WGLC (note mistake corrected in -09).
- * Clarified role of API for supporting full checksum.
- * Clarified that full checksum is required in security considerations, and therefore noting that full checksum should not be treated as an attack - consistent with remainder of document.
- * Added mention that API can set a mode in transport stack - to link to similar statement in RFC 2460 update.
- * Fixed typos.

Working Group Draft 10

- * Submission to correct unwanted removal of text from section 5 bullets 5-7 by GF.
- * Replaced section 5 text with the text from 08, and reapplied the editorial correction.
- * Note to reviewers: Please compare this revision with -08 used in the IETF LC).

Working Group Draft 11

- * Added REF for 5097 (Noted by S.Turner)
- * Added text in response to P. Resnick on place where checksum is calculated.
- * Added text to note experience with MPLS/PWE; Appendix updated to refer to this (S. Bryant)
- * Added text in response to P.Resnick's 2nd comments.
- * Request to make UDP-Lite more clearly recommended (J Touch, P.Resnick)

- * Added considerations around usage of zero checksum in routers.
- * Added text in response to Stewart Bryant's comments on router requirements.

Authors' Addresses

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

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

Magnus Westerlund
Ericsson
Farogatan 6
Stockholm, SE-164 80
Sweden

Phone: +46 8 719 0000
Email: magnus.westerlund@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 7, 2013

S. Shah
K. Patel
Cisco Systems
S. Bajaj
Juniper Networks
L. Tomotaki
Verizon
M. Boucadair
France Telecom
Jan 03, 2013

Inter-domain SLA Exchange
draft-ietf-idr-sla-exchange-00

Abstract

Network administrators typically provision QoS policies for their application traffic (such as voice, video) based on SLAs negotiated with their providers, and translate those SLAs to vendor specific configuration language. Both learning of SLA, either thru SLA documents or via some other out-of-band method, and translating them to vendor specific configuration language is a complex, many times manual, process and prone to errors. This document proposes an in-band method of SLA signaling which can help to simplify some of the complexities.

This document defines an operational transitive attribute to signal SLA details in-band, across administrative boundaries (considered as Autonomous Systems (AS)), and thus simplify/speed-up some of the complex tasks.

Though the use-case with the proposed attribute is explicitly defined in this document, purpose of this attribute is not limited to this use-case only.

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 7, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	4
2. Terminology	5
3. QoS Attribute Definition	6
3.1. SLA, QoS attribute sub-type, Definition	6
4. Originating SLA Notification	14
4.1. SLA Contexts	15
4.1.1. SLA advertisement for point to point connection	15
4.1.2. SLA advertisement for destination AS multiple hops away	15
5. SLA Attribute handling at forwarding nodes	16
5.1. BGP node capable of processing QoS attribute	16
5.2. BGP node not capable of processing QoS attribute	17
5.3. Aggregator	17
6. SLA attribute handling at Receiver	17
6.1. Traffic class mapping	18
7. Deployment Consideration	18
8. Acknowledgements	20
9. IANA Considerations	20
10. Security Considerations	21
11. References	22
11.1. Normative References	22
11.2. Informative References	22
Authors' Addresses	22

1. Introduction

Typically there is a contractual Service Level Agreement (SLA) negotiated between Customer and Provider or between one Provider to another Provider [CPP]. This contractual agreement defines the nature of the various traffic classes (i.e. traffic match conditions) and services needed for each traffic class. The contract may exist at different levels of traffic granularity. The contract could be full line-rate or sub rate for aggregate traffic. Or it could be even finer granular traffic distinction with services defined for standard code-points or for specific set of prefix or for set of well-known application types.

Once the SLA is negotiated, it needs to be translated into enforcing configuration data and policies on the Provider's Edge (PE) as well as on the Customer's Edge (CE). At the Customer, a person administering the CE device may be a different person, or even a different department, from the ones negotiating SLA contracts with the Provider and thus an administrator at the CE first requires to manually learn negotiated SLA, thru SLA documents or via some other off-band method. In a subsequent step an administrator requires to translate SLA to QoS policies using router (vendor) specific provisioning language. In a multi-vendor environment, translating the SLA into technology-specific configuration and then enforcing that configuration requires to consider specificities of each vendor. There does not exist any standard protocol to translate SLA agreements into technical clauses and configurations and thus both the steps of out of band learning of negotiated SLA and provisioning them in a vendor specific language can be complex and error-prone. For an example for voice service, the Provider may negotiate service for such traffic thru EF code-point in Diffserv networks. Administrator at the CE not only will have to know that Provider's service for voice traffic is EF based but will also have to implement DSCP EF classification rule along with Low Latency Service rule as per vendor's provisioning language.

Given the Provider also maintains established contracts, which very well may even be enforced at the PE, an in-band method of signaling it from the PE to the CE can help eliminate manual administrative process described above. Provider may have SLA negotiated with the Customer via some defined off-band method. Once negotiated, the Provider may translate that SLA in networking language on the PE (this process remains same as is done today). This SLA instance then can be signaled to the CE via some in-band protocol exchange. In reaction to that message, receiver CE router may automatically translate that to relevant QoS policy definition on the box. This in-band signaling method helps eliminate manual complex process required by administrator at the CE. Taking same voice service as an

example, given Provider already may provision definition of EF code-point for such, signaling this code-point traffic class from PE to CE along with low latency service definition, omits administrator at the CE to worry about such translation.

For in-band signaling, we propose use of BGP transport. The details of SLAs are independent of BGP and are specific to the granularity of traffic classes and their subsequent treatment. Though we find BGP as a suitable transport for inter-domain SLA exchange for the following reasons:

- The most common use-case of SLA exchange is across Autonomous Systems. And BGP is the most suitable protocol for any inter-domain exchange
- There is no other suitable protocol available today for SLA exchange
- BGP updates already advertise specific set of prefixes (flow or flow-group). Other QoS-related attributes, apart from the the use of SLA advertisement, can be added to these updates in the future

The proposal is a definition of a new BGP attribute to advertise/learn SLA details in-band. The BGP attribute proposed, in this document, is intended to advertise SLA from one AS to a list of interested AS. QoS services advertised could be for the incoming traffic to the AS community, that is advertising SLA or could be for the outgoing traffic from the advertiser or could be for both directions. Reception of and reaction to advertised SLAs are optional for the receiver.

The aim with the signaling of this attribute, across administrative boundaries, is to help network administrators speed up and simplify QoS provisioning with automatic learning of SLAs and thus avoiding complexities and possible errors with manual learning.

We propose QoS as an optional transitive attribute, keeping SLA advertisement and discovery (request) as one of the sub-types of QoS attribute. This is to keep QoS attribute open for extensions, in future, for other SLA specific requirements or even beyond SLA specific needs. For example, SLA Negotiation and Assurance is out of scope of this document which can be envisioned as another sub-type.

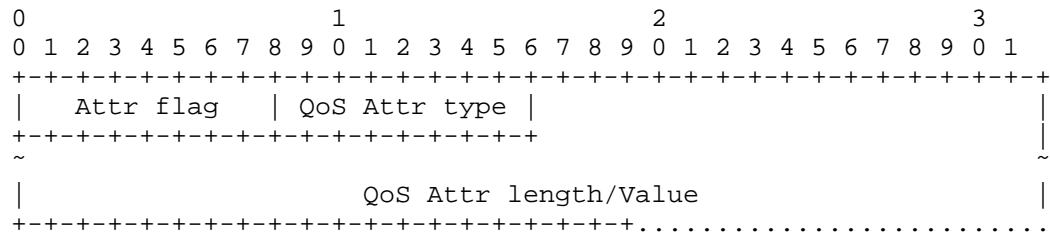
2. Terminology

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. QoS Attribute Definition

The QoS Attribute proposed, in BGP, is an optional transitive attribute (attribute type code to be assigned by IANA). SLA is defined as one of the sub-types in the QoS attribute.



Attribute flags

highest order bit (bit 0) -

MUST be set to 1, since this is an optional attribute

2nd higher order bit (bit 1) -

MUST be set to 1, since this is a transitive attribute

The first octet in the Value field of the QoS attribute is QoS Attribute specific flags

highest order bit (bit 0) -

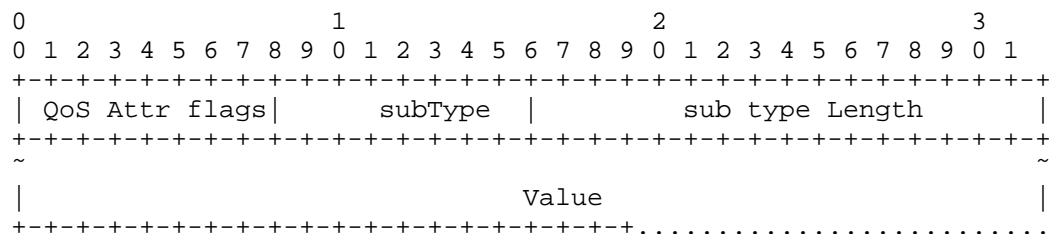
It defines if update message MUST be dropped (if set to 1) without updating routing data-base, when this is the last BGP receiver from the list of AS this attribute is announced to, or MUST announce (if set to 0) further to BGP peers

The purpose of this bit is discussed further in subsequent sections.

Remaining bits are currently unused and MUST be set to 0

3.1. SLA, QoS attribute sub-type, Definition

The value field of the QoS Attribute contains further TLVs, following QoS Attribute flags described in the previous section. One of the TLVs that we define is a tuple of (SLA sub-type, Length, Value)

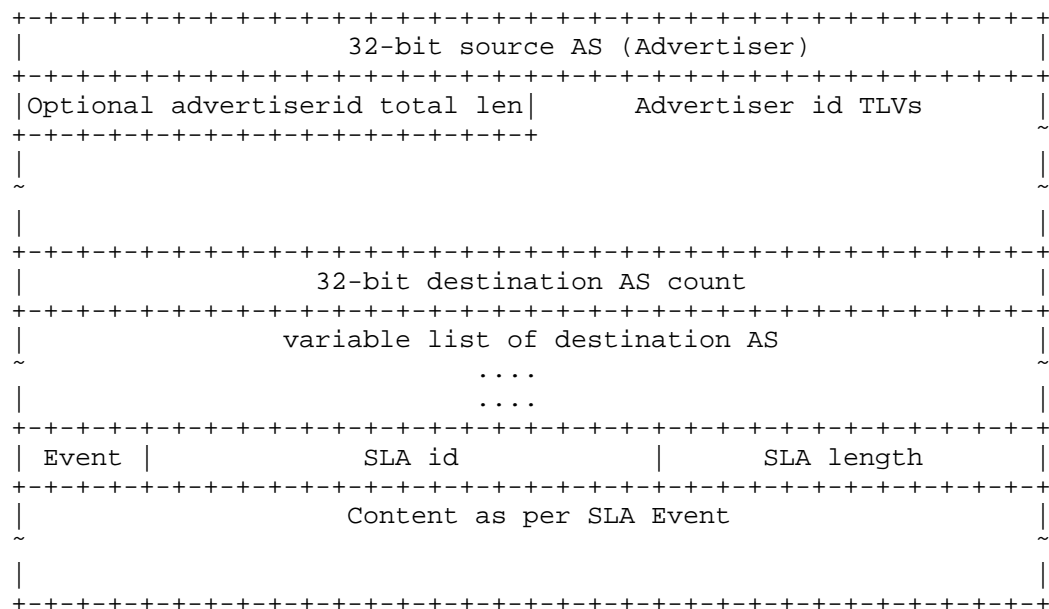


subType - 8 bits

0x00 = reserved
 0x01 = SLA
 0x02 - 0x0f = for future use

SLA sub-type specific value field details 1) sender and receiver(s) and 2) SLA parameters. SLA Parameters include SLA event type (such as Advertise, Request) and content associated to that event type.

The format of SLA message is,



Source AS

32-bit source AS number. This is the AS that is advertising SLA
0 = ignore Source and Destination AS list from this Value field.
Instead refer to Source and Destination AS as defined by BGP
message. SLA sub-type specifics, from the QoS attribute,
MUST be removed by the receiver in such case.

Optional advertiser id total len

16-bit Source address identifier (optional).

0 = No optional identifier

In general any additional qualifier for an advertiser is not
required. The SLA definition is in the context of prefix
advertised in the NLRI definition. The exception is where a BGP
speaker, in the middle of an update path to the destination AS,
aggregates prefixes. We will refer this middle BGP speaker, that
aggregates routes, as an Aggregator. Aggregator is then required
to insert original NLRI details in the optional advertiser field

Optional Advertiser id TLV

4-bit type

0x0 = reserved

0x1 = ORIGIN_NLRI, variable length

0x2 to 0xf = for future use,

Destination AS count

32-bit destination AS count to take variable length AS list.

This count has no functional value when Source AS is 0

0 = broadcast

Destination AS list

32-bit destination AS number, this field is omitted if broadcast

....

.... [as many as AS count]

....

SLA Event Type

4-bits

0x0 = reserved

0x1 = ADVERTISE

0x2 = REQUEST

0x3 to 0xf, for future use

SLA Id

16-bit identifier unique within the scope of source AS

The significance of an SLA identifier is in the context of the source that is advertising SLA. SLA identifier is not globally unique but it MUST be unique in the context of the source AS (advertiser).

The SLA content is optional for an advertised SLA id. If SLA content does not exist in BGP update messages with advertised SLA attribute then receiver MUST inherit prior advertised SLA content for the same SLA id from the same Source AS.

If advertised SLA id is different from earlier advertised one, for the same prefix, previous SLA MUST be replaced with the new advertised one.

SLA is aggregate for all the traffic to prefixes that share same source AS and SLA id.

SLA Length

12-bits

The format of SLA ADVERTISE event is,

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|dir|      Traffic Class count      | Class Desc Len|      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|      Traffic Class Description
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|      Traffic Class Elements count/values
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Service  Count|      service type/value pair
+---+---+---+---+---+
|
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|      Repeat from Traffic Class Description for next Traffic Class
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

|                                     |
| ~      Repeat from direction for SLA in the other direction      ~ |
|                                     |
+-----+

```

Direction

02-bit for incoming or outgoing traffic,
 0x0 = reserved
 0x1 = incoming, from destination AS towards source AS
 0x2 = outgoing, from source AS towards destination AS
 0x3 = for future use

Traffic Class count (Classifier Groups count)

16-bit, count of number of classifier groups
 00 = Advertisement to invalidate previous advertised SLA if was any

Traffic Class Descr Length

08-bit, size of the length

0 = No description

Traffic Class Description

Ascii Description of the Traffic Class

Traffic Class Elements Count in a Traffic Class,

08-bit count of classifier elements in a specific Traffic Class

00 = this has relative definition. It means classify rest all traffic that is not classified via earlier described Traffic Classes.
 It is RECOMMENDED to have 0 elements Traffic Class definition last in the ordered list. If Advertised SLA does not have this Traffic Class last in the advertised list, receivers MUST re-order it, for the forwarding purpose, as the last Traffic Class, in the ordered list, from the source AS. It is MUST that advertisement from a specific source does not have more than one Traffic classes with element count 0. If there are more than one such Traffic Classes then advertised SLA MUST be ignored. It is okay for SLA message though to have none Traffic Class with element count 0.

Classifier Element values in a Traffic Class (optional),

08-bit = type of the Element
variable-length = based on type of the Element

Element Types (08-bit)

0x00 = Invalid
0x01 = Reserved
0x02 = IP_DSCP, (length = 06-bits, value = 0..63)
0x03 = MPLS_TC, (length = 03-bits, value = 0..7)
0x04 = 802_1Q_COS, (length = 03-bits, value = 0..7)
0x05 = 802_1Q_DEI, (length = 01-bit, value = 0..1)
0x06 = PHB_ID, (length = 12-bits, value = 0..4095)
0x07 to 0xff = for future use

Traffic Class Service count (for a Traffic Class under definition)

08-bit count of service attributes fields to follow with
type/value pair

List of service types and relevant values are discussed below

00 = no bounded service (also means Best Effort)

Traffic Class Service (optional),

16-bit = type of the field
variable-length = based on type of the service

- 0x00 = reserved

- 0x01 = MINRATE

04-bit, unit type

0x00 = reserved

0x04 = PERCENT

0x05 = Kbps

0x06 to 0x0f = for future use

32-bit, value in unit kbps

- 0x02 = MINRATE_BURST

32-bit, value in bytes

- 0x03 = MINRATE_IN_PROFILE_MARKING
 - 04-bit, re-mark type
 - 0x00 = Invalid
 - 0x01 = Reserved
 - 0x02 = IP_DSCP
 - 0x03 = MPLS_TC
 - 0x04 = 802_1Q_COS
 - 0x05 = 802_1Q_DEI
 - 0x06 to 0x0f = for future use
 - 08-bit, value
- 0x04 = MINRATE_OUT_PROFILE_MARKING
 - 04-bit, re-mark type
 - 0x00 = Invalid
 - 0x01 = Reserved
 - 0x02 = IP_DSCP
 - 0x03 = MPLS_TC
 - 0x04 = 802_1Q_COS
 - 0x05 = 802_1Q_DEI
 - 0x06 to 0x0f = for future use
 - 08-bit, value
- 0x05 = MAXRATE
 - 04-bit, unit type
 - 0x00 = reserved
 - 0x04 = PERCENT
 - 0x05 = Kbps
 - 0x06 to 0x0f = for future use
 - 32-bit, value
- 0x06 = MAXRATE_BURST
 - 32-bit, value in bytes
- 0x07 = MAXRATE_IN_PROFILE_MARKING
 - 04-bit, re-mark type
 - 0x00 = Invalid
 - 0x01 = Reserved
 - 0x02 = IP_DSCP
 - 0x03 = MPLS_TC
 - 0x04 = 802_1Q_COS
 - 0x05 = 802_1Q_DEI
 - 0x06 to 0x0f = for future use
 - 08-bit, value

- 0x08 = MAXRATE_OUT_PROFILE_MARKING
 - 04-bit, re-mark type
 - 0x00 = Invalid
 - 0x01 = DROP
 - 0x02 = IP_DSCP
 - 0x03 = MPLS_TC
 - 0x04 = 802_1Q_COS
 - 0x05 = 802_1Q_DEI
 - 0x06 to 0x0f = for future use
 - 08-bit, value

In the case when MINRATE_IN_PROFILE_MARKING, MINRATE_OUT_PROFILE_MARKING, MAXRATE_IN_PROFILE_MARKING and MAXRATE_OUT_PROFILE_MARKING all of them are advertised,

- MINRATE_IN_PROFILE_MARKING takes highest precedence (that is over MAXRATE_IN_PROFILE_MARKING)
 - MAXRATE_IN_PROFILE_MARKING takes precedence over MINRATE_OUT_PROFILE_MARKING
 - and MAXRATE_OUT_PROFILE_MARKING takes precedence over MINRATE_OUT_PROFILE_MARKING
-
- 0x09 = DROP_THRESHOLD
 - 03-bit count of drop-priority fields to follow with (type,value, unit,value) tuple
 - 04-bit, drop priority type
 - 0x00 = Invalid
 - 0x01 = None
 - 0x02 = IP_DSCP
 - 0x03 = MPLS_EXP
 - 0x04 = 802_1Q_COS
 - 0x05 = 802_1Q_DEI
 - 0x06 to 0x0f = for future use
 - 08-bit, drop priority type value
 - 04-bit, unit type
 - 0x00 = reserved
 - 0x01 = TIME_US
 - 0x02 = PERCENT
 - 0x03 to 0x0f = for future use
 - 08-bit, drop threshold value as per unit type

- 0x0A = RELATIVE_PRIORITY
04-bit, priority value
lower the value, higher the priority

Relative priority indicates scheduling priority. For example voice traffic, that requires lowest latency compare to any other traffic, will have lowest value advertised in relative priority. For two different traffic classification groups where one application group may be considered more important than the other but from scheduling perspective do not require to be distinguish with different priority. Relative priority for those classification groups may be advertised with the same value.

- 0x0B = SUB_TRAFFIC_CLASSES
variable-length, repeats all content described above from Traffic Class count onwards.

For SLAs where a specific Traffic Class may further have differentiated services for sub-group of Classifier Elements, this service type SHOULD be used to further divide Traffic Class in multiple sub-classes. Each sub-class then defined with their own classifier elements and service types.

4. Originating SLA Notification

QoS attribute to advertise SLA MUST be added by the originator of a BGP UPDATE message. Any BGP speaker in the forwarding path of a message MUST NOT insert QoS attribute for the same prefix.

SLA messages SHOULD NOT be sent periodically just for the purpose of keep alive. Since SLA changes are in-frequent, some sort of SLA policy change can be considered as a trigger for the advertisement.

For any SLA modification, originator MUST re-advertise entire SLA. There is no provision to advertise partial SLA. To invalidate previously advertised SLA, a message MUST be sent with new SLA advertisement with Traffic Class count as 0.

4.1. SLA Contexts

In certain cases, the advertisement may be to establish SLA for aggregate traffic on a point to point connection between a specific destination and a specific source. A point to point connection may be a physical link, connecting BGP peers, or may be a virtual link (like tunnel). A BGP update message, in such cases, with source AS number and NLRI prefix of source end-point can uniquely identify physical/virtual link and so establishes advertised SLA's context for aggregate traffic for that point to point link.

In the simplest case where PE and CE are directly connected via a physical link and have only single link between them, CE can uniquely identify forwarding link to PE with AS number of the PE and NLRI prefix being an address of PE, to CE (that is next hop address from CE to PE). SLA advertised thru BGP update message from PE to CE, with PE's AS number and IP address, establishes SLA context for the aggregate traffic through link CE to PE. SLA advertised thru BGP update message from PE to CE, with PE's AS number and any other prefix establishes SLA for that specific prefix that is subset of traffic under CE to PE link.

Even though this example is in the context of IP prefix, SLA exchange does not have to be limited to IPv4 family only. SLA advertisement is generic to all forms of NLRI types that are supported by the BGP protocol specification (like IPV4, IPV6, VPN-IPV4, VPN-IPV6).

4.1.1. SLA advertisement for point to point connection

When SLA messages are intended to be advertised for the point to point connection (physical or logical), the message is destined for the next hop and advertised message is in the context of the prefix of the source end-point of the point to point connection.

The destination AS number set to, within QoS SLA attribute, typically is of the neighbor BGP speaker's. Alternatively, originator MAY not encode source/destination AS numbers (that is source AS set to 0 and destination AS count set to 0), in the QoS attribute. The most significant bit of the QoS attribute flag MAY be set to 1, specifically it MUST be set to 1 when intention is to not install route update, at the receiver, for the advertised message.

4.1.2. SLA advertisement for destination AS multiple hops away

When SLA messages are to be advertised beyond next hop, value of source AS, in the QoS attribute, MUST be set by the originator of the update message. If such update is meant to be for specific list of AS(es) as receiver then list of destination AS MUST be populated in

the QoS attribute message to avoid flooding of the QoS attribute data in the network beyond those destinations.

When a new prefix is added in the AS, AS for which SLA has already been advertised before for other existing prefixes, then to advertise that new prefix to be part of earlier advertised SLA, a trigger of new BGP update message with QoS attribute containing SLA id is sufficient. Update message does not require to have whole SLA content.

When BGP update messages are triggered as a result of SLA policy change and so for the purpose of SLA exchange only, forwarding BGP update messages beyond intended receivers are not necessary. Highest order bit in the QoS Attribute flag MUST be set to suggest receiver to drop entire BGP update message [Note that it is an indication to drop entire update message, not only QoS attribute], after all intended receivers have processed it. If update message contains list of destination of AS then message MUST be dropped only after all intended receivers (destinations) have received it.

5. SLA Attribute handling at forwarding nodes

5.1. BGP node capable of processing QoS attribute

If a BGP node is capable of processing QoS attribute, it optionally MAY process the message. If advertised SLA has list of destination AS, it MAY trim list and so count of destination AS to exclude ones that are not required in further announcement of BGP updates.

BGP node MUST drop SLA related sub type from the QoS attribute, if none of the AS from the destination list is in the forwarding path. Rest of the QoS attributes message MAY be forwarded if there exist other sub-types of QoS attribute and forwarding rules meets other sub-types requirements. If there is no other sub-types existing in the QoS attribute message then node MUST drop QoS attribute all together. Rest other attributes and NLRI may be announced further if it meets rules defined by other attributes and BGP protocol.

If most significant bit in the QoS attribute flag is set to 1 then entire BGP update message MUST be dropped if there are no destination left in the list to advertise to. However, If SLA message is meant to be broadcast then message MUST not be dropped/trimmed.

Except extracting entire SLA sub-type of the QoS attribute, trimming the list of destination AS list and inserting NLRI at Aggregator

node, rest all other content MUST not be modified by any intermediate receivers of the message.

5.2. BGP node not capable of processing QoS attribute

If BGP node is not capable of processing QoS attribute, it MUST forward attribute message as it is received.

5.3. Aggregator

It is RECOMMENDED to not aggregate prefixes from BGP update messages that contain QoS SLA attribute. If Aggregator MUST aggregate prefixes then it MUST copy QoS SLA attribute in new aggregated BGP update message. At the same time, it MUST also insert NLRI, from the original update message, as an optional advertiser id to go along with source AS in the QoS attribute.

To support SLA exchange multiple hops away in the path that has one of the forwarding node in the path acting as Aggregator, it is required Aggregator node to be capable of processing QoS attribute.

6. SLA attribute handling at Receiver

Reception of and reaction to advertised messages are optional for the receiver.

As described in earlier section, while reacting to SLA advertisement - receiver SHOULD invalidate previous advertised SLA and then if one exists for advertised NLRI. If new advertised SLA update is with non-zero Traffic Class count, new advertised SLA SHOULD be installed. If new advertised SLA update is with Traffic Class count 0, no action is required.

- If advertised QoS Attribute is with flag set to indicate to drop this message, receiver MUST drop message if it is the last receiver, in the update path, this message is advertised to.

If advertised SLA is from the next hop, in reverse path, the receiver can establish advertised SLA for the whole link, the link could be physical or virtual link, associated with the next hop. If NLRI advertised in update message is not of the next hop, receiver may establish advertised SLA for that specific prefix list under the relevant link. It is completely up to the receiver to decide for which prefixes to accept advertised SLA and for which ones to not.

For cases where if earlier message has not yet reached to the intended receiver, a re-signaling is required. A signaling event

REQUEST is required, for this purpose, to be triggered by intended receiver. Since BGP messages are considered reliable, discussion of REQUEST, for this purpose or any other purpose, is considered out of the scope of this document.

To handle error conditions, the approach of "attribute-discard" as mentioned in [IDR-ERR] MAY be used in an event if a QoS attribute parsing results in any attribute errors. Alternatively, an approach of "treat-as-withdraw" MAY be used as mentioned in [IDR-ERR] if an implementation also wishes to withdraw the associated prefix.

6.1. Traffic class mapping

It is common that switching/routing technologies used in 2 different AS could be different. For example, Provider may tunnel Customer's IP traffic thru MPLS cloud. In such cases traffic class definition for QoS services is also different in both AS. For the meaningful use of advertised SLA in such cases, receiver is required to map traffic class from one type to another.

In the example given, traffic classification in Customer AS could be IP Diffserv based whereas traffic classification in Provider AS could be MPLS TC based. Thus for advertised MPLS TC based SLA from PE, CE would require to map traffic class from IP Diffserv based to MPLS TC type.

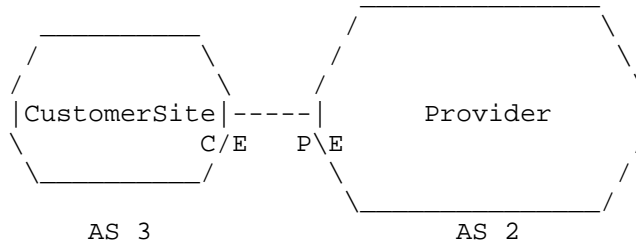
There are well-defined recommendations that exist for traffic class mapping between two technologies. Receiver MAY use those defined recommendations for traffic class mapping or MAY define its own as per its network Traffic Class service definition to map to advertised Traffic Classes. It is completely up to the receiver how to define such traffic class mapping.

7. Deployment Consideration

Typical use-case aimed with this proposal is for Provider to advertise contracted SLA to Customer Edge. SLA established between customer and Provider is provisioned by the provider on the PE device (facing Customer Edge). This provisioning, in a form supported by Provider, is advertised thru proposed BGP QoS attribute to the Customer Edge. Customer may read thru advertised SLA to provision one on the Customer Edge link facing towards PE.

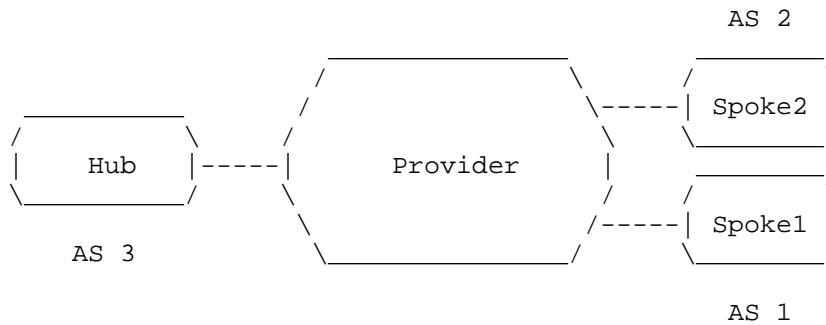
Contracted SLA from PE to CE may be full line-rate or sub-rate of a link or finer granular controlled services. SLA is not required to be advertised if the SLA contract is simply a physical link. SLA advertise can be useful when contracted service is sub-rate of a link

and/or if for finer granular traffic classes that are controlled. Like voice, video services may be capped to certain rate.



SLA_ADVERTISE: AS2 to AS3
NLRI = PE ip address

Another use-case can be to advertise SLA among different network sites within one Enterprise network. In Hub and Spoke deployments, Hub may define SLA for individual spokes and advertise this SLA thru BGP updates.



SLA_ADVERTISE: AS2 to AS3
NLRI = AS2 tunnel address

SLA_ADVERTISE: AS1 to AS3
NLRI = AS2 tunnel address

It very well could be possible that AS2 may first learn its SLA with Provider from Provider Edge it is connected to and then advertises

same or subset of the SLA to AS3 with AS2 to AS3 tunnel's ip address as NLRI.

Deployment options are not limited to involving CEs only. For any contract between Provider to Provider, SLA may be advertised from one PE to another PE also.

8. Acknowledgements

Thanks to Fred Baker for his suggestions and to Ken Briley, Rahul Patel, Fred Yip, Lou Berger and Brian Carpenter for the review. Thanks to Bertrand Duvivier for his valuable contributions to help make subsequent revision better.

9. IANA Considerations

This document defines a new BGP attribute. IANA maintains the list of existing BGP attribute types. Proposal is to define a new attribute type code for the QoS attribute.

With the proposal, there is a list defined for Traffic Class Elements type and associated Service types. IANA will be required to maintain list of both new types.

Proposed definition of Traffic Class Element Types

0x00 = Invalid
0x01 = Reserved
0x02 = IP_DSCP, (length = 06-bits, value = 0..63)
0x03 = MPLS_TC, (length = 03-bits, value = 0..7)
0x04 = 802_1Q_COS, (length = 03-bits, value = 0..7)
0x05 = 802_1Q_DEI, (length = 01-bit, value = 0..1)
0x06 = PHB_ID, (length = 12-bits, value = 0..4095)

Proposed definition of Traffic Class Service Types

0x00 = reserved
0x01 = MINRATE
0x02 = MINRATE_BURST
0x03 = MINRATE_IN_PROFILE_MARKING
0x04 = MINRATE_OUT_PROFILE_MARKING
0x05 = MAXRATE
0x06 = MAXRATE_BURST
0x07 = MAXRATE_IN_PROFILE_MARKING
0x08 = MAXRATE_OUT_PROFILE_MARKING
0x09 = DROP_THRESHOLD
0x0A = RELATIVE_PRIORITY
0x0B = SUB_TRAFFIC_CLASSES

Proposed definition of Unit Types

0x00 = reserved
0x01 = TIME_US
0x02 = PERCENT
0x03 = Kbps

10. Security Considerations

There is a potential for mis-behaved AS to advertise wrong SLA, stealing identity of another AS. This resembles to problems already identified and resolved, in the routing world, thru reverse path forwarding check. One proposal, inline to RPF, to resolve such threats is to have each BGP speaker node, in the forwarding path, perform reverse path check on source AS.

Since we expect these messages to originate and distributed in the managed network, there should not be any risks for identity theft. Thus reverse path check is not considered in this proposal nor have we considered any alternates. Such solutions can be explored later if any such need.

11. References

11.1. Normative References

- [RFC1771] Rekhter, Y. and T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, March 1995.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.
- [RFC3140] Black, D., Brim, S., Carpenter, B., and F. Le Faucheur, "Per Hop Behavior Identification Codes", RFC 3140, June 2001.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.
- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.
- [RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, February 2006.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.
- [IDR-ERR] Scudder, J., Chen, E., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Message, I-D.draft-ietf-idr-error-handling", June 2012.

11.2. Informative References

- [CPP] Boucadair, M., Jacquenet, C., and N. Wang, "IP/MPLS Connectivity Provisioning Profile, I-D.boucadair-connectivity-provisioning-profile", Sep 2012.

Authors' Addresses

Shitanshu Shah
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
US

Email: svshah@cisco.com

Keyur Patel
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
US

Email: keyupate@cisco.com

Sandeep Bajaj
Juniper Networks
1194 N. Mathilda Avenue
Sunnyvale, CA 94089
US

Email: sbajaj@juniper.net

Luis Tomotaki
Verizon
400 International
Richardson, TX 75081
US

Email: luis.tomotaki@verizon.com

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Transport Area Working Group
Internet-Draft
Updates: 2309 (if approved)
Intended status: BCP
Expires: May 11, 2014

B. Briscoe
BT
J. Manner
Aalto University
November 07, 2013

Byte and Packet Congestion Notification
draft-ietf-tsvwg-byte-pkt-congest-12

Abstract

This document provides recommendations of best current practice for dropping or marking packets using any active queue management (AQM) algorithm, including random early detection (RED), BLUE, pre-congestion notification (PCN) and newer schemes such as CoDel (Controlled Delay) and PIE (Proportional Integral controller Enhanced). We give three strong recommendations: (1) packet size should be taken into account when transports detect and respond to congestion indications, (2) packet size should not be taken into account when network equipment creates congestion signals (marking, dropping), and therefore (3) in the specific case of RED, the byte-mode packet drop variant that drops fewer small packets should not be used. This memo updates RFC 2309 to deprecate deliberate preferential treatment of small packets in AQM algorithms.

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 11, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
1.1. Terminology and Scoping	6
1.2. Example Comparing Packet-Mode Drop and Byte-Mode Drop	7
2. Recommendations	9
2.1. Recommendation on Queue Measurement	9
2.2. Recommendation on Encoding Congestion Notification	10
2.3. Recommendation on Responding to Congestion	11
2.4. Recommendation on Handling Congestion Indications when Splitting or Merging Packets	12
3. Motivating Arguments	12
3.1. Avoiding Perverse Incentives to (Ab)use Smaller Packets	12
3.2. Small != Control	14
3.3. Transport-Independent Network	14
3.4. Partial Deployment of AQM	15
3.5. Implementation Efficiency	17
4. A Survey and Critique of Past Advice	17
4.1. Congestion Measurement Advice	18
4.1.1. Fixed Size Packet Buffers	18
4.1.2. Congestion Measurement without a Queue	19
4.2. Congestion Notification Advice	20
4.2.1. Network Bias when Encoding	20
4.2.2. Transport Bias when Decoding	22
4.2.3. Making Transports Robust against Control Packet Losses	23
4.2.4. Congestion Notification: Summary of Conflicting Advice	24
5. Outstanding Issues and Next Steps	25
5.1. Bit-congestible Network	25
5.2. Bit- & Packet-congestible Network	25
6. Security Considerations	26
7. IANA Considerations	26
8. Conclusions	26
9. Acknowledgements	28
10. Comments Solicited	28
11. References	28
11.1. Normative References	28
11.2. Informative References	28
Appendix A. Survey of RED Implementation Status	32
Appendix B. Sufficiency of Packet-Mode Drop	34
B.1. Packet-Size (In)Dependence in Transports	35
B.2. Bit-Congestible and Packet-Congestible Indications	38
Appendix C. Byte-mode Drop Complicates Policing Congestion Response	39
Appendix D. Changes from Previous Versions	40

1. Introduction

This document provides recommendations of best current practice for how we should correctly scale congestion control functions with respect to packet size for the long term. It also recognises that expediency may be necessary to deal with existing widely deployed protocols that don't live up to the long term goal.

When signalling congestion, the problem of how (and whether) to take packet sizes into account has exercised the minds of researchers and practitioners for as long as active queue management (AQM) has been discussed. Indeed, one reason AQM was originally introduced was to reduce the lock-out effects that small packets can have on large packets in drop-tail queues. This memo aims to state the principles we should be using and to outline how these principles will affect future protocol design, taking into account the existing deployments we have already.

The question of whether to take into account packet size arises at three stages in the congestion notification process:

Measuring congestion: When a congested resource measures locally how congested it is, should it measure its queue length in time, bytes or packets?

Encoding congestion notification into the wire protocol: When a congested network resource signals its level of congestion, should it drop / mark each packet dependent on the size of the particular packet in question?

Decoding congestion notification from the wire protocol: When a transport interprets the notification in order to decide how much to respond to congestion, should it take into account the size of each missing or marked packet?

Consensus has emerged over the years concerning the first stage, which Section 2.1 records in the RFC Series. In summary: If possible it is best to measure congestion by time in the queue, but otherwise the choice between bytes and packets solely depends on whether the resource is congested by bytes or packets.

The controversy is mainly around the last two stages: whether to allow for the size of the specific packet notifying congestion i) when the network encodes or ii) when the transport decodes the congestion notification.

Currently, the RFC series is silent on this matter other than a paper trail of advice referenced from [RFC2309], which conditionally

recommends byte-mode (packet-size dependent) drop [pktByteEmail]. Reducing drop of small packets certainly has some tempting advantages: i) it drops less control packets, which tend to be small and ii) it makes TCP's bit-rate less dependent on packet size. However, there are ways of addressing these issues at the transport layer, rather than reverse engineering network forwarding to fix the problems.

This memo updates [RFC2309] to deprecate deliberate preferential treatment of packets in AQM algorithms solely because of their size. It recommends that (1) packet size should be taken into account when transports detect and respond to congestion indications, (2) not when network equipment creates them. This memo also adds to the congestion control principles enumerated in BCP 41 [RFC2914].

In the particular case of Random early Detection (RED), this means that the byte-mode packet drop variant should not be used to drop fewer small packets, because that creates a perverse incentive for transports to use tiny segments, consequently also opening up a DoS vulnerability. Fortunately all the RED implementers who responded to our admittedly limited survey (Section 4.2.4) have not followed the earlier advice to use byte-mode drop, so the position this memo argues for seems to already exist in implementations.

However, at the transport layer, TCP congestion control is a widely deployed protocol that doesn't scale with packet size (i.e. its reduction in rate does not take into account the size of a lost packet). To date this hasn't been a significant problem because most TCP implementations have been used with similar packet sizes. But, as we design new congestion control mechanisms, this memo recommends that we should build in scaling with packet size rather than assuming we should follow TCP's example.

This memo continues as follows. First it discusses terminology and scoping. Section 2 gives the concrete formal recommendations, followed by motivating arguments in Section 3. We then critically survey the advice given previously in the RFC series and the research literature (Section 4), referring to an assessment of whether or not this advice has been followed in production networks (Appendix A). To wrap up, outstanding issues are discussed that will need resolution both to inform future protocol designs and to handle legacy (Section 5). Then security issues are collected together in Section 6 before conclusions are drawn in Section 8. The interested reader can find discussion of more detailed issues on the theme of byte vs. packet in the appendices.

This memo intentionally includes a non-negligible amount of material on the subject. For the busy reader Section 2 summarises the

recommendations for the Internet community.

1.1. Terminology and Scoping

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].

This memo applies to the design of all AQM algorithms, for example, Random Early Detection (RED) [RFC2309], BLUE [BLUE02], Pre-Congestion Notification (PCN) [RFC5670], Controlled Delay (CoDel) [I-D.nichols-tsvwg-codel] and the Proportional Integral controller Enhanced (PIE) [I-D.pan-tsvwg-pie]. Throughout, RED is used as a concrete example because it is a widely known and deployed AQM algorithm. There is no intention to imply that the advice is any less applicable to the other algorithms, nor that RED is preferred.

Congestion Notification: Congestion notification is a changing signal that aims to communicate the probability that the network resource(s) will not be able to forward the level of traffic load offered (or that there is an impending risk that they will not be able to).

The 'impending risk' qualifier is added, because AQM systems set a virtual limit smaller than the actual limit to the resource, then notify when this virtual limit is exceeded in order to avoid uncontrolled congestion of the actual capacity.

Congestion notification communicates a real number bounded by the range [0 , 1]. This ties in with the most well-understood measure of congestion notification: drop probability.

Explicit and Implicit Notification: The byte vs. packet dilemma concerns congestion notification irrespective of whether it is signalled implicitly by drop or using Explicit Congestion Notification (ECN [RFC3168] or PCN [RFC5670]). Throughout this document, unless clear from the context, the term marking will be used to mean notifying congestion explicitly, while congestion notification will be used to mean notifying congestion either implicitly by drop or explicitly by marking.

Bit-congestible vs. Packet-congestible: If the load on a resource depends on the rate at which packets arrive, it is called packet-congestible. If the load depends on the rate at which bits arrive it is called bit-congestible.

Examples of packet-congestible resources are route look-up engines and firewalls, because load depends on how many packet headers

they have to process. Examples of bit-congestible resources are transmission links, radio power and most buffer memory, because the load depends on how many bits they have to transmit or store. Some machine architectures use fixed size packet buffers, so buffer memory in these cases is packet-congestible (see Section 4.1.1).

The path through a machine will typically encounter both packet-congestible and bit-congestible resources. However, currently, a design goal of network processing equipment such as routers and firewalls is to size the packet-processing engine(s) relative to the lines in order to keep packet processing uncongested even under worst case packet rates with runs of minimum size packets. Therefore, packet-congestion is currently rare [RFC6077; S.3.3], but there is no guarantee that it will not become more common in future.

Note that information is generally processed or transmitted with a minimum granularity greater than a bit (e.g. octets). The appropriate granularity for the resource in question should be used, but for the sake of brevity we will talk in terms of bytes in this memo.

Coarser Granularity: Resources may be congestible at higher levels of granularity than bits or packets, for instance stateful firewalls are flow-congestible and call-servers are session-congestible. This memo focuses on congestion of connectionless resources, but the same principles may be applicable for congestion notification protocols controlling per-flow and per-session processing or state.

RED Terminology: In RED whether to use packets or bytes when measuring queues is called respectively "packet-mode queue measurement" or "byte-mode queue measurement". And whether the probability of dropping a particular packet is independent or dependent on its size is called respectively "packet-mode drop" or "byte-mode drop". The terms byte-mode and packet-mode should not be used without specifying whether they apply to queue measurement or to drop.

1.2. Example Comparing Packet-Mode Drop and Byte-Mode Drop

Taking RED as a well-known example algorithm, a central question addressed by this document is whether to recommend RED's packet-mode drop variant and to deprecate byte-mode drop. Table 1 compares how packet-mode and byte-mode drop affect two flows of different size packets. For each it gives the expected number of packets and of bits dropped in one second. Each example flow runs at the same bit-

rate of 48Mb/s, but one is broken up into small 60 byte packets and the other into large 1500 byte packets.

To keep up the same bit-rate, in one second there are about 25 times more small packets because they are 25 times smaller. As can be seen from the table, the packet rate is 100,000 small packets versus 4,000 large packets per second (pps).

Parameter	Formula	Small packets	Large packets
Packet size	$s/8$	60B	1,500B
Packet size	s	480b	12,000b
Bit-rate	x	48Mbps	48Mbps
Packet-rate	$u = x/s$	100kpps	4kpps
Packet-mode Drop			
Pkt loss probability	p	0.1%	0.1%
Pkt loss-rate	$p*u$	100pps	4pps
Bit loss-rate	$p*u*s$	48kbps	48kbps
Byte-mode Drop			
	MTU, $M=12,000b$		
Pkt loss probability	$b = p*s/M$	0.004%	0.1%
Pkt loss-rate	$b*u$	4pps	4pps
Bit loss-rate	$b*u*s$	1.92kbps	48kbps

Table 1: Example Comparing Packet-mode and Byte-mode Drop

For packet-mode drop, we illustrate the effect of a drop probability of 0.1%, which the algorithm applies to all packets irrespective of size. Because there are 25 times more small packets in one second, it naturally drops 25 times more small packets, that is 100 small packets but only 4 large packets. But if we count how many bits it drops, there are 48,000 bits in 100 small packets and 48,000 bits in 4 large packets--the same number of bits of small packets as large.

The packet-mode drop algorithm drops any bit with the same probability whether the bit is in a small or a large packet.

For byte-mode drop, again we use an example drop probability of 0.1%, but only for maximum size packets (assuming the link maximum transmission unit (MTU) is 1,500B or 12,000b). The byte-mode algorithm reduces the drop probability of smaller packets proportional to their size, making the probability that it drops a small packet 25 times smaller at 0.004%. But there are 25 times more small packets, so dropping them with 25 times lower probability results in dropping the same number of packets: 4 drops in both cases. The 4 small dropped packets contain 25 times less bits than the 4 large dropped packets: 1,920 compared to 48,000.

The byte-mode drop algorithm drops any bit with a probability proportionate to the size of the packet it is in.

2. Recommendations

This section gives recommendations related to network equipment in Sections 2.1 and 2.2, and in Sections 2.3 and 2.4 we discuss the implications on the transport protocols.

2.1. Recommendation on Queue Measurement

Ideally, an AQM would measure the service time of the queue to measure congestion of a resource. However service time can only be measured as packets leave the queue, where it is not always expedient to implement a full AQM algorithm. To predict the service time as packets join the queue, an AQM algorithm needs to measure the length of the queue.

In this case, if the resource is bit-congestible, the AQM implementation SHOULD measure the length of the queue in bytes and, if the resource is packet-congestible, the implementation SHOULD measure the length of the queue in packets. Subject to the exceptions below, no other choice makes sense, because the number of packets waiting in the queue isn't relevant if the resource gets congested by bytes and vice versa. For example, the length of the queue into a transmission line would be measured in bytes, while the length of the queue into a firewall would be measured in packets.

To avoid the pathological effects of drop tail, the AQM can then transform this service time or queue length into the probability of dropping or marking a packet (e.g. RED's piecewise linear function between thresholds).

What this advice means for RED as a specific example:

1. A RED implementation SHOULD use byte mode queue measurement for measuring the congestion of bit-congestible resources and packet mode queue measurement for packet-congestible resources.
2. An implementation SHOULD NOT make it possible to configure the way a queue measures itself, because whether a queue is bit-congestible or packet-congestible is an inherent property of the queue.

Exceptions to these recommendations might be necessary, for instance where a packet-congestible resource has to be configured as a proxy bottleneck for a bit-congestible resource in an adjacent box that does not support AQM.

The recommended approach in less straightforward scenarios, such as fixed size packet buffers, resources without a queue and buffers comprising a mix of packet and bit-congestible resources, is discussed in Section 4.1. For instance, Section 4.1.1 explains that the queue into a line should be measured in bytes even if the queue consists of fixed-size packet-buffers, because the root-cause of any congestion is bytes arriving too fast for the line--packets filling buffers are merely a symptom of the underlying congestion of the line.

2.2. Recommendation on Encoding Congestion Notification

When encoding congestion notification (e.g. by drop, ECN or PCN), the probability that network equipment drops or marks a particular packet to notify congestion SHOULD NOT depend on the size of the packet in question. As the example in Section 1.2 illustrates, to drop any bit with probability 0.1% it is only necessary to drop every packet with probability 0.1% without regard to the size of each packet.

This approach ensures the network layer offers sufficient congestion information for all known and future transport protocols and also ensures no perverse incentives are created that would encourage transports to use inappropriately small packet sizes.

What this advice means for RED as a specific example:

1. The RED AQM algorithm SHOULD NOT use byte-mode drop, i.e. it ought to use packet-mode drop. Byte-mode drop is more complex, it creates the perverse incentive to fragment segments into tiny pieces and it is vulnerable to floods of small packets.
2. If a vendor has implemented byte-mode drop, and an operator has turned it on, it is RECOMMENDED to switch it to packet-mode drop, after establishing if there are any implications on the relative performance of applications using different packet sizes. The unlikely possibility of some application-specific legacy use of byte-mode drop is the only reason that all the above recommendations on encoding congestion notification are not phrased more strongly.

RED as a whole SHOULD NOT be switched off. Without RED, a drop tail queue biases against large packets and is vulnerable to floods of small packets.

Note well that RED's byte-mode queue drop is completely orthogonal to byte-mode queue measurement and should not be confused with it. If a RED implementation has a byte-mode but does not specify what sort of byte-mode, it is most probably byte-mode queue measurement, which is

fine. However, if in doubt, the vendor should be consulted.

A survey (Appendix A) showed that there appears to be little, if any, installed base of the byte-mode drop variant of RED. This suggests that deprecating byte-mode drop will have little, if any, incremental deployment impact.

2.3. Recommendation on Responding to Congestion

When a transport detects that a packet has been lost or congestion marked, it SHOULD consider the strength of the congestion indication as proportionate to the size in octets (bytes) of the missing or marked packet.

In other words, when a packet indicates congestion (by being lost or marked) it can be considered conceptually as if there is a congestion indication on every octet of the packet, not just one indication per packet.

To be clear, the above recommendation solely describes how a transport should interpret the meaning of a congestion indication, as a long term goal. It makes no recommendation on whether a transport should act differently based on this interpretation. It merely aids interoperability between transports, if they choose to make their actions depend on the strength of congestion indications.

This definition will be useful as the IETF transport area continues its programme of;

- o updating host-based congestion control protocols to take account of packet size
- o making transports less sensitive to losing control packets like SYNs and pure ACKs.

What this advice means for the case of TCP:

1. If two TCP flows with different packet sizes are required to run at equal bit rates under the same path conditions, this SHOULD be done by altering TCP (Section 4.2.2), not network equipment (the latter affects other transports besides TCP).
2. If it is desired to improve TCP performance by reducing the chance that a SYN or a pure ACK will be dropped, this SHOULD be done by modifying TCP (Section 4.2.3), not network equipment.

To be clear, we are not recommending at all that TCPs under equivalent conditions should aim for equal bit-rates. We are merely

saying that anyone trying to do such a thing should modify their TCP algorithm, not the network.

These recommendations are phrased as 'SHOULD' rather than 'MUST', because there may be cases where expediency dictates that compatibility with pre-existing versions of a transport protocol make the recommendations impractical.

2.4. Recommendation on Handling Congestion Indications when Splitting or Merging Packets

Packets carrying congestion indications may be split or merged in some circumstances (e.g. at a RTP/RTCP transcoder or during IP fragment reassembly). Splitting and merging only make sense in the context of ECN, not loss.

The general rule to follow is that the number of octets in packets with congestion indications SHOULD be equivalent before and after merging or splitting. This is based on the principle used above; that an indication of congestion on a packet can be considered as an indication of congestion on each octet of the packet.

The above rule is not phrased with the word "MUST" to allow the following exception. There are cases where pre-existing protocols were not designed to conserve congestion marked octets (e.g. IP fragment reassembly [RFC3168] or loss statistics in RTCP receiver reports [RFC3550] before ECN was added [RFC6679]). When any such protocol is updated, it SHOULD comply with the above rule to conserve marked octets. However, the rule may be relaxed if it would otherwise become too complex to interoperate with pre-existing implementations of the protocol.

One can think of a splitting or merging process as if all the incoming congestion-marked octets increment a counter and all the outgoing marked octets decrement the same counter. In order to ensure that congestion indications remain timely, even the smallest positive remainder in the conceptual counter should trigger the next outgoing packet to be marked (causing the counter to go negative).

3. Motivating Arguments

This section is informative. It justifies the recommendations given in the previous section.

3.1. Avoiding Perverse Incentives to (Ab)use Smaller Packets

Increasingly, it is being recognised that a protocol design must take care not to cause unintended consequences by giving the parties in

the protocol exchange perverse incentives [Evol_cc][RFC3426]. Given there are many good reasons why larger path maximum transmission units (PMTUs) would help solve a number of scaling issues, we do not want to create any bias against large packets that is greater than their true cost.

Imagine a scenario where the same bit rate of packets will contribute the same to bit-congestion of a link irrespective of whether it is sent as fewer larger packets or more smaller packets. A protocol design that caused larger packets to be more likely to be dropped than smaller ones would be dangerous in both the following cases:

Malicious transports: A queue that gives an advantage to small packets can be used to amplify the force of a flooding attack. By sending a flood of small packets, the attacker can get the queue to discard more traffic in large packets, allowing more attack traffic to get through to cause further damage. Such a queue allows attack traffic to have a disproportionately large effect on regular traffic without the attacker having to do much work.

Non-malicious transports: Even if an application designer is not actually malicious, if over time it is noticed that small packets tend to go faster, designers will act in their own interest and use smaller packets. Queues that give advantage to small packets create an evolutionary pressure for applications or transports to send at the same bit-rate but break their data stream down into tiny segments to reduce their drop rate. Encouraging a high volume of tiny packets might in turn unnecessarily overload a completely unrelated part of the system, perhaps more limited by header-processing than bandwidth.

Imagine two unresponsive flows arrive at a bit-congestible transmission link each with the same bit rate, say 1Mbps, but one consists of 1500B and the other 60B packets, which are 25x smaller. Consider a scenario where gentle RED [gentle_RED] is used, along with the variant of RED we advise against, i.e. where the RED algorithm is configured to adjust the drop probability of packets in proportion to each packet's size (byte mode packet drop). In this case, RED aims to drop 25x more of the larger packets than the smaller ones. Thus, for example if RED drops 25% of the larger packets, it will aim to drop 1% of the smaller packets (but in practice it may drop more as congestion increases [RFC4828; Appx B.4]). Even though both flows arrive with the same bit rate, the bit rate the RED queue aims to pass to the line will be 750kbps for the flow of larger packets but 990kbps for the smaller packets (because of rate variations it will actually be a little less than this target).

Note that, although the byte-mode drop variant of RED amplifies small

packet attacks, drop-tail queues amplify small packet attacks even more (see Security Considerations in Section 6). Wherever possible neither should be used.

3.2. Small != Control

Dropping fewer control packets considerably improves performance. It is tempting to drop small packets with lower probability in order to improve performance, because many control packets tend to be smaller (TCP SYNs & ACKs, DNS queries & responses, SIP messages, HTTP GETs, etc). However, we must not give control packets preference purely by virtue of their smallness, otherwise it is too easy for any data source to get the same preferential treatment simply by sending data in smaller packets. Again we should not create perverse incentives to favour small packets rather than to favour control packets, which is what we intend.

Just because many control packets are small does not mean all small packets are control packets.

So, rather than fix these problems in the network, we argue that the transport should be made more robust against losses of control packets (see 'Making Transports Robust against Control Packet Losses' in Section 4.2.3).

3.3. Transport-Independent Network

TCP congestion control ensures that flows competing for the same resource each maintain the same number of segments in flight, irrespective of segment size. So under similar conditions, flows with different segment sizes will get different bit-rates.

To counter this effect it seems tempting not to follow our recommendation, and instead for the network to bias congestion notification by packet size in order to equalise the bit-rates of flows with different packet sizes. However, in order to do this, the queuing algorithm has to make assumptions about the transport, which become embedded in the network. Specifically:

- o The queuing algorithm has to assume how aggressively the transport will respond to congestion (see Section 4.2.4). If the network assumes the transport responds as aggressively as TCP NewReno, it will be wrong for Compound TCP and differently wrong for Cubic TCP, etc. To achieve equal bit-rates, each transport then has to guess what assumption the network made, and work out how to replace this assumed aggressiveness with its own aggressiveness.

- o Also, if the network biases congestion notification by packet size it has to assume a baseline packet size--all proposed algorithms use the local MTU (for example see the byte-mode loss probability formula in Table 1). Then if the non-Reno transports mentioned above are trying to reverse engineer what the network assumed, they also have to guess the MTU of the congested link.

Even though reducing the drop probability of small packets (e.g. RED's byte-mode drop) helps ensure TCP flows with different packet sizes will achieve similar bit rates, we argue this correction should be made to any future transport protocols based on TCP, not to the network in order to fix one transport, no matter how predominant it is. Effectively, favouring small packets is reverse engineering of network equipment around one particular transport protocol (TCP), contrary to the excellent advice in [RFC3426], which asks designers to question "Why are you proposing a solution at this layer of the protocol stack, rather than at another layer?"

In contrast, if the network never takes account of packet size, the transport can be certain it will never need to guess any assumptions the network has made. And the network passes two pieces of information to the transport that are sufficient in all cases: i) congestion notification on the packet and ii) the size of the packet. Both are available for the transport to combine (by taking account of packet size when responding to congestion) or not. Appendix B checks that these two pieces of information are sufficient for all relevant scenarios.

When the network does not take account of packet size, it allows transport protocols to choose whether to take account of packet size or not. However, if the network were to bias congestion notification by packet size, transport protocols would have no choice; those that did not take account of packet size themselves would unwittingly become dependent on packet size, and those that already took account of packet size would end up taking account of it twice.

3.4. Partial Deployment of AQM

In overview, the argument in this section runs as follows:

- o Because the network does not and cannot always drop packets in proportion to their size, it shouldn't be given the task of making drop signals depend on packet size at all.
- o Transports on the other hand don't always want to make their rate response proportional to the size of dropped packets, but if they want to, they always can.

The argument is similar to the end-to-end argument that says "Don't do X in the network if end-systems can do X by themselves, and they want to be able to choose whether to do X anyway." Actually the following argument is stronger; in addition it says "Don't give the network task X that could be done by the end-systems, if X is not deployed on all network nodes, and end-systems won't be able to tell whether their network is doing X, or whether they need to do X themselves." In this case, the X in question is "making the response to congestion depend on packet size".

We will now re-run this argument taking each step in more depth. The argument applies solely to drop, not to ECN marking.

A queue drops packets for either of two reasons: a) to signal to host congestion controls that they should reduce the load and b) because there is no buffer left to store the packets. Active queue management tries to use drops as a signal for hosts to slow down (case a) so that drop due to buffer exhaustion (case b) should not be necessary.

AQM is not universally deployed in every queue in the Internet; many cheap Ethernet bridges, software firewalls, NATs on consumer devices, etc implement simple tail-drop buffers. Even if AQM were universal, it has to be able to cope with buffer exhaustion (by switching to a behaviour like tail-drop), in order to cope with unresponsive or excessive transports. For these reasons networks will sometimes be dropping packets as a last resort (case b) rather than under AQM control (case a).

When buffers are exhausted (case b), they don't naturally drop packets in proportion to their size. The network can only reduce the probability of dropping smaller packets if it has enough space to store them somewhere while it waits for a larger packet that it can drop. If the buffer is exhausted, it does not have this choice. Admittedly tail-drop does naturally drop somewhat fewer small packets, but exactly how few depends more on the mix of sizes than the size of the packet in question. Nonetheless, in general, if we wanted networks to do size-dependent drop, we would need universal deployment of (packet-size dependent) AQM code, which is currently unrealistic.

A host transport cannot know whether any particular drop was a deliberate signal from an AQM or a sign of a queue shedding packets due to buffer exhaustion. Therefore, because the network cannot universally do size-dependent drop, it should not do it all.

Whereas universality is desirable in the network, diversity is desirable between different transport layer protocols - some, like

NewReno TCP [RFC5681], may not choose to make their rate response proportionate to the size of each dropped packet, while others will (e.g. TFRC-SP [RFC4828]).

3.5. Implementation Efficiency

Biasing against large packets typically requires an extra multiply and divide in the network (see the example byte-mode drop formula in Table 1). Allowing for packet size at the transport rather than in the network ensures that neither the network nor the transport needs to do a multiply operation--multiplication by packet size is effectively achieved as a repeated add when the transport adds to its count of marked bytes as each congestion event is fed to it. Also the work to do the biasing is spread over many hosts, rather than concentrated in just the congested network element. These aren't principled reasons in themselves, but they are a happy consequence of the other principled reasons.

4. A Survey and Critique of Past Advice

This section is informative, not normative.

The original 1993 paper on RED [RED93] proposed two options for the RED active queue management algorithm: packet mode and byte mode. Packet mode measured the queue length in packets and dropped (or marked) individual packets with a probability independent of their size. Byte mode measured the queue length in bytes and marked an individual packet with probability in proportion to its size (relative to the maximum packet size). In the paper's outline of further work, it was stated that no recommendation had been made on whether the queue size should be measured in bytes or packets, but noted that the difference could be significant.

When RED was recommended for general deployment in 1998 [RFC2309], the two modes were mentioned implying the choice between them was a question of performance, referring to a 1997 email [pktByteEmail] for advice on tuning. A later addendum to this email introduced the insight that there are in fact two orthogonal choices:

- o whether to measure queue length in bytes or packets (Section 4.1)
- o whether the drop probability of an individual packet should depend on its own size (Section 4.2).

The rest of this section is structured accordingly.

4.1. Congestion Measurement Advice

The choice of which metric to use to measure queue length was left open in RFC2309. It is now well understood that queues for bit-congestible resources should be measured in bytes, and queues for packet-congestible resources should be measured in packets [pktByteEmail].

Congestion in some legacy bit-congestible buffers is only measured in packets not bytes. In such cases, the operator has to set the thresholds mindful of a typical mix of packets sizes. Any AQM algorithm on such a buffer will be oversensitive to high proportions of small packets, e.g. a DoS attack, and under-sensitive to high proportions of large packets. However, there is no need to make allowances for the possibility of such legacy in future protocol design. This is safe because any under-sensitivity during unusual traffic mixes cannot lead to congestion collapse given the buffer will eventually revert to tail drop, discarding proportionately more large packets.

4.1.1. Fixed Size Packet Buffers

The question of whether to measure queues in bytes or packets seems to be well understood. However, measuring congestion is confusing when the resource is bit congestible but the queue into the resource is packet congestible. This section outlines the approach to take.

Some, mostly older, queuing hardware allocates fixed sized buffers in which to store each packet in the queue. This hardware forwards to the line in one of two ways:

- o With some hardware, any fixed sized buffers not completely filled by a packet are padded when transmitted to the wire. This case, should clearly be treated as packet-congestible, because both queuing and transmission are in fixed MTU-sized units. Therefore the queue length in packets is a good model of congestion of the link.
- o More commonly, hardware with fixed size packet buffers transmits packets to line without padding. This implies a hybrid forwarding system with transmission congestion dependent on the size of packets but queue congestion dependent on the number of packets, irrespective of their size.

Nonetheless, there would be no queue at all unless the line had become congested--the root-cause of any congestion is too many bytes arriving for the line. Therefore, the AQM should measure the queue length as the sum of all the packet sizes in bytes that

are queued up waiting to be serviced by the line, irrespective of whether each packet is held in a fixed size buffer.

In the (unlikely) first case where use of padding means the queue should be measured in packets, further confusion is likely because the fixed buffers are rarely all one size. Typically pools of different sized buffers are provided (Cisco uses the term 'buffer carving' for the process of dividing up memory into these pools [IOSArch]). Usually, if the pool of small buffers is exhausted, arriving small packets can borrow space in the pool of large buffers, but not vice versa. However, there is no need to consider all this complexity, because the root-cause of any congestion is still line overload--buffer consumption is only the symptom. Therefore, the length of the queue should be measured as the sum of the bytes in the queue that will be transmitted to line, including any padding. In the (unusual) case of transmission with padding this means the sum of the sizes of the small buffers queued plus the sum of the sizes of the large buffers queued.

We will return to borrowing of fixed sized buffers when we discuss biasing the drop/marketing probability of a specific packet because of its size in Section 4.2.1. But here we can repeat the simple rule for how to measure the length of queues of fixed buffers: no matter how complicated the buffering scheme is, ultimately a transmission line is nearly always bit-congestible so the number of bytes queued up waiting for the line measures how congested the line is, and it is rarely important to measure how congested the buffering system is.

4.1.1.2. Congestion Measurement without a Queue

AQM algorithms are nearly always described assuming there is a queue for a congested resource and the algorithm can use the queue length to determine the probability that it will drop or mark each packet. But not all congested resources lead to queues. For instance, power limited resources are usually bit-congestible if energy is primarily required for transmission rather than header processing, but it is rare for a link protocol to build a queue as it approaches maximum power.

Nonetheless, AQM algorithms do not require a queue in order to work. For instance spectrum congestion can be modelled by signal quality using target bit-energy-to-noise-density ratio. And, to model radio power exhaustion, transmission power levels can be measured and compared to the maximum power available. [ECNFixedWireless] proposes a practical and theoretically sound way to combine congestion notification for different bit-congestible resources at different layers along an end to end path, whether wireless or wired, and whether with or without queues.

In wireless protocols that use request to send / clear to send (RTS / CTS) control, such as some variants of IEEE802.11, it is reasonable to base an AQM on the time spent waiting for transmission opportunities (TXOPs) even though wireless spectrum is usually regarded as congested by bits (for a given coding scheme). This is because requests for TXOPs queue up as the spectrum gets congested by all the bits being transferred. So the time that TXOPs are queued directly reflects bit congestion of the spectrum.

4.2. Congestion Notification Advice

4.2.1. Network Bias when Encoding

4.2.1.1. Advice on Packet Size Bias in RED

The previously mentioned email [pktByteEmail] referred to by [RFC2309] advised that most scarce resources in the Internet were bit-congestible, which is still believed to be true (Section 1.1). But it went on to offer advice that is updated by this memo. It said that drop probability should depend on the size of the packet being considered for drop if the resource is bit-congestible, but not if it is packet-congestible. The argument continued that if packet drops were inflated by packet size (byte-mode dropping), "a flow's fraction of the packet drops is then a good indication of that flow's fraction of the link bandwidth in bits per second". This was consistent with a referenced policing mechanism being worked on at the time for detecting unusually high bandwidth flows, eventually published in 1999 [pBox]. However, the problem could and should have been solved by making the policing mechanism count the volume of bytes randomly dropped, not the number of packets.

A few months before RFC2309 was published, an addendum was added to the above archived email referenced from the RFC, in which the final paragraph seemed to partially retract what had previously been said. It clarified that the question of whether the probability of dropping/markings a packet should depend on its size was not related to whether the resource itself was bit congestible, but a completely orthogonal question. However the only example given had the queue measured in packets but packet drop depended on the size of the packet in question. No example was given the other way round.

In 2000, Cnodder et al [REDbyte] pointed out that there was an error in the part of the original 1993 RED algorithm that aimed to distribute drops uniformly, because it didn't correctly take into account the adjustment for packet size. They recommended an algorithm called RED_4 to fix this. But they also recommended a further change, RED_5, to adjust drop rate dependent on the square of relative packet size. This was indeed consistent with one implied

motivation behind RED's byte mode drop--that we should reverse engineer the network to improve the performance of dominant end-to-end congestion control mechanisms. This memo makes a different recommendations in Section 2.

By 2003, a further change had been made to the adjustment for packet size, this time in the RED algorithm of the ns2 simulator. Instead of taking each packet's size relative to a 'maximum packet size' it was taken relative to a 'mean packet size', intended to be a static value representative of the 'typical' packet size on the link. We have not been able to find a justification in the literature for this change, however Eddy and Allman conducted experiments [REDbias] that assessed how sensitive RED was to this parameter, amongst other things. However, this changed algorithm can often lead to drop probabilities of greater than 1 (which gives a hint that there is probably a mistake in the theory somewhere).

On 10-Nov-2004, this variant of byte-mode packet drop was made the default in the ns2 simulator. It seems unlikely that byte-mode drop has ever been implemented in production networks (Appendix A), therefore any conclusions based on ns2 simulations that use RED without disabling byte-mode drop are likely to behave very differently from RED in production networks.

4.2.1.2. Packet Size Bias Regardless of AQM

The byte-mode drop variant of RED (or a similar variant of other AQM algorithms) is not the only possible bias towards small packets in queueing systems. We have already mentioned that tail-drop queues naturally tend to lock-out large packets once they are full.

But also queues with fixed sized buffers reduce the probability that small packets will be dropped if (and only if) they allow small packets to borrow buffers from the pools for larger packets (see Section 4.1.1). Borrowing effectively makes the maximum queue size for small packets greater than that for large packets, because more buffers can be used by small packets while less will fit large packets. Incidentally, the bias towards small packets from buffer borrowing is nothing like as large as that of RED's byte-mode drop.

Nonetheless, fixed-buffer memory with tail drop is still prone to lock-out large packets, purely because of the tail-drop aspect. So, fixed size packet-buffers should be augmented with a good AQM algorithm and packet-mode drop. If an AQM is too complicated to implement with multiple fixed buffer pools, the minimum necessary to prevent large packet lock-out is to ensure smaller packets never use the last available buffer in any of the pools for larger packets.

4.2.2. Transport Bias when Decoding

The above proposals to alter the network equipment to bias towards smaller packets have largely carried on outside the IETF process. Whereas, within the IETF, there are many different proposals to alter transport protocols to achieve the same goals, i.e. either to make the flow bit-rate take account of packet size, or to protect control packets from loss. This memo argues that altering transport protocols is the more principled approach.

A recently approved experimental RFC adapts its transport layer protocol to take account of packet sizes relative to typical TCP packet sizes. This proposes a new small-packet variant of TCP-friendly rate control [RFC5348] called TFRC-SP [RFC4828]. Essentially, it proposes a rate equation that inflates the flow rate by the ratio of a typical TCP segment size (1500B including TCP header) over the actual segment size [PktSizeEquCC]. (There are also other important differences of detail relative to TFRC, such as using virtual packets [CCvarPktSize] to avoid responding to multiple losses per round trip and using a minimum inter-packet interval.)

Section 4.5.1 of this TFRC-SP spec discusses the implications of operating in an environment where queues have been configured to drop smaller packets with proportionately lower probability than larger ones. But it only discusses TCP operating in such an environment, only mentioning TFRC-SP briefly when discussing how to define fairness with TCP. And it only discusses the byte-mode dropping version of RED as it was before Cnoddler et al pointed out it didn't sufficiently bias towards small packets to make TCP independent of packet size.

So the TFRC-SP spec doesn't address the issue of which of the network or the transport should handle fairness between different packet sizes. In its Appendix B.4 it discusses the possibility of both TFRC-SP and some network buffers duplicating each other's attempts to deliberately bias towards small packets. But the discussion is not conclusive, instead reporting simulations of many of the possibilities in order to assess performance but not recommending any particular course of action.

The paper originally proposing TFRC with virtual packets (VP-TFRC) [CCvarPktSize] proposed that there should perhaps be two variants to cater for the different variants of RED. However, as the TFRC-SP authors point out, there is no way for a transport to know whether some queues on its path have deployed RED with byte-mode packet drop (except if an exhaustive survey found that no-one has deployed it!--see Appendix A). Incidentally, VP-TFRC also proposed that byte-mode RED dropping should really square the packet-size compensation-factor

(like that of Cnoder's RED_5, but apparently unaware of it).

Pre-congestion notification [RFC5670] is an IETF technology to use a virtual queue for AQM marking for packets within one Diffserv class in order to give early warning prior to any real queuing. The PCN marking algorithms have been designed not to take account of packet size when forwarding through queues. Instead the general principle has been to take account of the sizes of marked packets when monitoring the fraction of marking at the edge of the network, as recommended here.

4.2.3. Making Transports Robust against Control Packet Losses

Recently, two RFCs have defined changes to TCP that make it more robust against losing small control packets [RFC5562] [RFC5690]. In both cases they note that the case for these two TCP changes would be weaker if RED were biased against dropping small packets. We argue here that these two proposals are a safer and more principled way to achieve TCP performance improvements than reverse engineering RED to benefit TCP.

Although there are no known proposals, it would also be possible and perfectly valid to make control packets robust against drop by requesting a scheduling class with lower drop probability, by re-marking to a Diffserv code point [RFC2474] within the same behaviour aggregate.

Although not brought to the IETF, a simple proposal from Wischik [DupTCP] suggests that the first three packets of every TCP flow should be routinely duplicated after a short delay. It shows that this would greatly improve the chances of short flows completing quickly, but it would hardly increase traffic levels on the Internet, because Internet bytes have always been concentrated in the large flows. It further shows that the performance of many typical applications depends on completion of long serial chains of short messages. It argues that, given most of the value people get from the Internet is concentrated within short flows, this simple expedient would greatly increase the value of the best efforts Internet at minimal cost. A similar but more extensive approach has been evaluated on Google servers [GentleAggro].

The proposals discussed in this sub-section are experimental approaches that are not yet in wide operational use, but they are existence proofs that transports can make themselves robust against loss of control packets. The examples are all TCP-based, but applications over non-TCP transports could mitigate loss of control packets by making similar use of Diffserv, data duplication, FEC etc.

4.2.4. Congestion Notification: Summary of Conflicting Advice

transport cc	RED_1 (packet mode drop)	RED_4 (linear byte mode drop)	RED_5 (square byte mode drop)
TCP or TFRC	s/\sqrt{p}	$\sqrt{s/p}$	$1/\sqrt{p}$
TFRC-SP	$1/\sqrt{p}$	$1/\sqrt{sp}$	$1/(s.\sqrt{p})$

Table 2: Dependence of flow bit-rate per RTT on packet size, s , and drop probability, p , when network and/or transport bias towards small packets to varying degrees

Table 2 aims to summarise the potential effects of all the advice from different sources. Each column shows a different possible AQM behaviour in different queues in the network, using the terminology of Cnoder et al outlined earlier (RED_1 is basic RED with packet-mode drop). Each row shows a different transport behaviour: TCP [RFC5681] and TFRC [RFC5348] on the top row with TFRC-SP [RFC4828] below. Each cell shows how the bits per round trip of a flow depends on packet size, s , and drop probability, p . In order to declutter the formulae to focus on packet-size dependence they are all given per round trip, which removes any RTT term.

Let us assume that the goal is for the bit-rate of a flow to be independent of packet size. Suppressing all inessential details, the table shows that this should either be achievable by not altering the TCP transport in a RED_5 network, or using the small packet TFRC-SP transport (or similar) in a network without any byte-mode dropping RED (top right and bottom left). Top left is the 'do nothing' scenario, while bottom right is the 'do-both' scenario in which bit-rate would become far too biased towards small packets. Of course, if any form of byte-mode dropping RED has been deployed on a subset of queues that congest, each path through the network will present a different hybrid scenario to its transport.

Whatever, we can see that the linear byte-mode drop column in the middle would considerably complicate the Internet. It's a half-way house that doesn't bias enough towards small packets even if one believes the network should be doing the biasing. Section 2 recommends that all bias in network equipment towards small packets should be turned off--if indeed any equipment vendors have implemented it--leaving packet-size bias solely as the preserve of the transport layer (solely the leftmost, packet-mode drop column).

In practice it seems that no deliberate bias towards small packets

has been implemented for production networks. Of the 19% of vendors who responded to a survey of 84 equipment vendors, none had implemented byte-mode drop in RED (see Appendix A for details).

5. Outstanding Issues and Next Steps

5.1. Bit-congestible Network

For a connectionless network with nearly all resources being bit-congestible the recommended position is clear--that the network should not make allowance for packet sizes and the transport should. This leaves two outstanding issues:

- o How to handle any legacy of AQM with byte-mode drop already deployed;
- o The need to start a programme to update transport congestion control protocol standards to take account of packet size.

A survey of equipment vendors (Section 4.2.4) found no evidence that byte-mode packet drop had been implemented, so deployment will be sparse at best. A migration strategy is not really needed to remove an algorithm that may not even be deployed.

A programme of experimental updates to take account of packet size in transport congestion control protocols has already started with TFRC-SP [RFC4828].

5.2. Bit- & Packet-congestible Network

The position is much less clear-cut if the Internet becomes populated by a more even mix of both packet-congestible and bit-congestible resources (see Appendix B.2). This problem is not pressing, because most Internet resources are designed to be bit-congestible before packet processing starts to congest (see Section 1.1).

The IRTF Internet congestion control research group (ICCRG) has set itself the task of reaching consensus on generic forwarding mechanisms that are necessary and sufficient to support the Internet's future congestion control requirements (the first challenge in [RFC6077]). The research question of whether packet congestion might become common and what to do if it does may in the future be explored in the IRTF (the "Challenge 3: Packet Size" in [RFC6077]).

Note that sometimes it seems that resources might be congested by neither bits nor packets, e.g. where the queue for access to a wireless medium is in units of transmission opportunities. However,

the root cause of congestion of the underlying spectrum is overload of bits (see Section 4.1.2).

6. Security Considerations

This memo recommends that queues do not bias drop probability due to packets size. For instance dropping small packets less often than large creates a perverse incentive for transports to break down their flows into tiny segments. One of the benefits of implementing AQM was meant to be to remove this perverse incentive that drop-tail queues gave to small packets.

In practice, transports cannot all be trusted to respond to congestion. So another reason for recommending that queues do not bias drop probability towards small packets is to avoid the vulnerability to small packet DDoS attacks that would otherwise result. One of the benefits of implementing AQM was meant to be to remove drop-tail's DoS vulnerability to small packets, so we shouldn't add it back again.

If most queues implemented AQM with byte-mode drop, the resulting network would amplify the potency of a small packet DDoS attack. At the first queue the stream of packets would push aside a greater proportion of large packets, so more of the small packets would survive to attack the next queue. Thus a flood of small packets would continue on towards the destination, pushing regular traffic with large packets out of the way in one queue after the next, but suffering much less drop itself.

Appendix C explains why the ability of networks to police the response of any transport to congestion depends on bit-congestible network resources only doing packet-mode not byte-mode drop. In summary, it says that making drop probability depend on the size of the packets that bits happen to be divided into simply encourages the bits to be divided into smaller packets. Byte-mode drop would therefore irreversibly complicate any attempt to fix the Internet's incentive structures.

7. IANA Considerations

This document has no actions for IANA.

8. Conclusions

This memo identifies the three distinct stages of the congestion notification process where implementations need to decide whether to take packet size into account. The recommendations provided in Section 2 of this memo are different in each case:

- o When network equipment measures the length of a queue, if it is not feasible to use time it is recommended to count in bytes if the network resource is congested by bytes, or to count in packets if is congested by packets.
- o When network equipment decides whether to drop (or mark) a packet, it is recommended that the size of the particular packet should not be taken into account
- o However, when a transport algorithm responds to a dropped or marked packet, the size of the rate reduction should be proportionate to the size of the packet.

In summary, the answers are 'it depends', 'no' and 'yes' respectively

For the specific case of RED, this means that byte-mode queue measurement will often be appropriate but the use of byte-mode drop is very strongly discouraged.

At the transport layer the IETF should continue updating congestion control protocols to take account of the size of each packet that indicates congestion. Also the IETF should continue to make protocols less sensitive to losing control packets like SYN's, pure ACKs and DNS exchanges. Although many control packets happen to be small, the alternative of network equipment favouring all small packets would be dangerous. That would create perverse incentives to split data transfers into smaller packets.

The memo develops these recommendations from principled arguments concerning scaling, layering, incentives, inherent efficiency, security and policeability. But it also addresses practical issues such as specific buffer architectures and incremental deployment. Indeed a limited survey of RED implementations is discussed, which shows there appears to be little, if any, installed base of RED's byte-mode drop. Therefore it can be deprecated with little, if any, incremental deployment complications.

The recommendations have been developed on the well-founded basis that most Internet resources are bit-congestible not packet-congestible. We need to know the likelihood that this assumption will prevail longer term and, if it might not, what protocol changes will be needed to cater for a mix of the two. The IRTF Internet Congestion Control Research Group (ICCRG) is currently working on these problems [RFC6077].

9. Acknowledgements

Thank you to Sally Floyd, who gave extensive and useful review comments. Also thanks for the reviews from Philip Eardley, David Black, Fred Baker, David Taht, Toby Moncaster, Arnaud Jacquet and Mirja Kuehlewind as well as helpful explanations of different hardware approaches from Larry Dunn and Fred Baker. We are grateful to Bruce Davie and his colleagues for providing a timely and efficient survey of RED implementation in Cisco's product range. Also grateful thanks to Toby Moncaster, Will Dormann, John Regnault, Simon Carter and Stefaan De Cnodder who further helped survey the current status of RED implementation and deployment and, finally, thanks to the anonymous individuals who responded.

Bob Briscoe and Jukka Manner were partly funded by Trilogy, a research project (ICT- 216372) supported by the European Community under its Seventh Framework Programme. The views expressed here are those of the authors only.

10. 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.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

11.2. Informative References

- [BLUE02] Feng, W-c., Shin, K., Kandlur, D., and D. Saha, "The BLUE active queue management algorithms", IEEE/ACM Transactions on Networking 10(4) 513--528, August 2002, <<http://dx.doi.org/10.1109/TNET.2002.801399>>.
- [CCvarPktSize] Widmer, J., Boutremans, C., and J-Y. Le

- Boudec, "Congestion Control for Flows with Variable Packet Size", ACM CCR 34(2) 137--151, 2004, <<http://doi.acm.org/10.1145/997150.997162>>.
- [CHOke_Var_Pkt] Psounis, K., Pan, R., and B. Prabhaker, "Approximate Fair Dropping for Variable Length Packets", IEEE Micro 21(1):48--56, January-February 2001, <<http://www.stanford.edu/~balaji/papers/01approximatefair.pdf>>.
- [DRQ] Shin, M., Chong, S., and I. Rhee, "Dual-Resource TCP/AQM for Processing-Constrained Networks", IEEE/ACM Transactions on Networking Vol 16, issue 2, April 2008, <<http://dx.doi.org/10.1109/TNET.2007.900415>>.
- [DupTCP] Wischik, D., "Short messages", Philosophical Transactions of the Royal Society A 366(1872):1941-1953, June 2008, <<http://rsta.royalsocietypublishing.org/content/366/1872/1941.full.pdf+html>>.
- [ECNFixedWireless] Siris, V., "Resource Control for Elastic Traffic in CDMA Networks", Proc. ACM MOBICOM'02 , September 2002, <http://www.ics.forth.gr/netlab/publications/resource_control_elastic_cdma.html>.
- [Evol_cc] Gibbens, R. and F. Kelly, "Resource pricing and the evolution of congestion control", Automatica 35(12):1969--1985, December 1999, <<http://www.statslab.cam.ac.uk/~frank/evol.html>>.
- [GentleAggro] Flach, T., Dukkupati, N., Terzis, A., Raghavan, B., Cardwell, N., Cheng, Y., Jain, A., Hao, S., Katz-Bassett, E., and R. Govindan, "Reducing Web Latency: the Virtue of Gentle Aggression", ACM SIGCOMM CCR 43(4):159--170, August 2013, <<http://doi.acm.org/10.1145/2486001.2486014>>.
- [I-D.nichols-tsvwg-codel] Nichols, K. and V. Jacobson, "Controlled Delay Active Queue Management",

- draft-nichols-tsvwg-codel-01 (work in progress), February 2013.
- [I-D.pan-tsvwg-pie] Pan, R., Natarajan, P., Piglione, C., and M. Prabhu, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem", draft-pan-tsvwg-pie-00 (work in progress), December 2012.
- [IOSArch] Bollapragada, V., White, R., and C. Murphy, "Inside Cisco IOS Software Architecture", Cisco Press: CCIE Professional Development ISBN13: 978-1-57870-181-0, July 2000.
- [PktSizeEquCC] Vasallo, P., "Variable Packet Size Equation-Based Congestion Control", ICSI Technical Report tr-00-008, 2000, <<http://http.icsi.berkeley.edu/ftp/global/pub/techreports/2000/tr-00-008.pdf>>.
- [RED93] Floyd, S. and V. Jacobson, "Random Early Detection (RED) gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking 1(4) 397--413, August 1993, <<http://www.icir.org/floyd/papers/red/red.html>>.
- [REDBias] Eddy, W. and M. Allman, "A Comparison of RED's Byte and Packet Modes", Computer Networks 42(3) 261--280, June 2003, <<http://www.ir.bbn.com/documents/articles/redbias.ps>>.
- [REDbyte] De Cnodder, S., Elloumi, O., and K. Pauwels, "RED behavior with different packet sizes", Proc. 5th IEEE Symposium on Computers and Communications (ISCC) 793--799, July 2000, <<http://www.icir.org/floyd/red/Elloumi99.pdf>>.
- [RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet",

RFC 2309, April 1998.

- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.
- [RFC3426] Floyd, S., "General Architectural and Policy Considerations", RFC 3426, November 2002.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3714] Floyd, S. and J. Kempf, "IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet", RFC 3714, March 2004.
- [RFC4828] Floyd, S. and E. Kohler, "TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant", RFC 4828, April 2007.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.
- [RFC5562] Kuzmanovic, A., Mondal, A., Floyd, S., and K. Ramakrishnan, "Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets", RFC 5562, June 2009.
- [RFC5670] Eardley, P., "Metering and Marking Behaviour of PCN-Nodes", RFC 5670, November 2009.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

- [RFC5690] Floyd, S., Arcia, A., Ros, D., and J. Iyengar, "Adding Acknowledgement Congestion Control to TCP", RFC 5690, February 2010.
- [RFC6077] Papadimitriou, D., Welzl, M., Scharf, M., and B. Briscoe, "Open Research Issues in Internet Congestion Control", RFC 6077, February 2011.
- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.
- [RFC6789] Briscoe, B., Woundy, R., and A. Cooper, "Congestion Exposure (ConEx) Concepts and Use Cases", RFC 6789, December 2012.
- [Rate_fair_Dis] Briscoe, B., "Flow Rate Fairness: Dismantling a Religion", ACM CCR 37(2)63--74, April 2007, <<http://portal.acm.org/citation.cfm?id=1232926>>.
- [gentle_RED] Floyd, S., "Recommendation on using the "gentle_" variant of RED", Web page , March 2000, <<http://www.icir.org/floyd/red/gentle.html>>.
- [pBox] Floyd, S. and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", IEEE/ACM Transactions on Networking 7(4) 458--472, August 1999, <<http://www.aciri.org/floyd/end2end-paper.html>>.
- [pktByteEmail] Floyd, S., "RED: Discussions of Byte and Packet Modes", email , March 1997, <<http://www-nrg.ee.lbl.gov/floyd/REDaveraging.txt>>.

Appendix A. Survey of RED Implementation Status

This Appendix is informative, not normative.

In May 2007 a survey was conducted of 84 vendors to assess how widely drop probability based on packet size has been implemented in RED Table 3. About 19% of those surveyed replied, giving a sample size

of 16. Although in most cases we do not have permission to identify the respondents, we can say that those that have responded include most of the larger equipment vendors, covering a large fraction of the market. The two who gave permission to be identified were Cisco and Alcatel-Lucent. The others range across the large network equipment vendors at L3 & L2, firewall vendors, wireless equipment vendors, as well as large software businesses with a small selection of networking products. All those who responded confirmed that they have not implemented the variant of RED with drop dependent on packet size (2 were fairly sure they had not but needed to check more thoroughly). At the time the survey was conducted, Linux did not implement RED with packet-size bias of drop, although we have not investigated a wider range of open source code.

Response	No. of vendors	%age of vendors
Not implemented	14	17%
Not implemented (probably)	2	2%
Implemented	0	0%
No response	68	81%
Total companies/orgs surveyed	84	100%

Table 3: Vendor Survey on byte-mode drop variant of RED (lower drop probability for small packets)

Where reasons have been given, the extra complexity of packet bias code has been most prevalent, though one vendor had a more principled reason for avoiding it--similar to the argument of this document.

Our survey was of vendor implementations, so we cannot be certain about operator deployment. But we believe many queues in the Internet are still tail-drop. The company of one of the co-authors (BT) has widely deployed RED, but many tail-drop queues are bound to still exist, particularly in access network equipment and on middleboxes like firewalls, where RED is not always available.

Routers using a memory architecture based on fixed size buffers with borrowing may also still be prevalent in the Internet. As explained in Section 4.2.1, these also provide a marginal (but legitimate) bias towards small packets. So even though RED byte-mode drop is not prevalent, it is likely there is still some bias towards small packets in the Internet due to tail drop and fixed buffer borrowing.

Appendix B. Sufficiency of Packet-Mode Drop

This Appendix is informative, not normative.

Here we check that packet-mode drop (or marking) in the network gives sufficiently generic information for the transport layer to use. We check against a 2x2 matrix of four scenarios that may occur now or in the future (Table 4). The horizontal and vertical dimensions have been chosen because each tests extremes of sensitivity to packet size in the transport and in the network respectively.

Note that this section does not consider byte-mode drop at all. Having deprecated byte-mode drop, the goal here is to check that packet-mode drop will be sufficient in all cases.

Network	Transport	a) Independent of packet size of congestion notifications	b) Dependent on packet size of congestion notifications
1) Predominantly bit-congestible network		Scenario a1)	Scenario b1)
2) Mix of bit-congestible and pkt-congestible network		Scenario a2)	Scenario b2)

Table 4: Four Possible Congestion Scenarios

Appendix B.1 focuses on the horizontal dimension of Table 4 checking that packet-mode drop (or marking) gives sufficient information, whether or not the transport uses it--scenarios b) and a) respectively.

Appendix B.2 focuses on the vertical dimension of Table 4, checking that packet-mode drop gives sufficient information to the transport whether resources in the network are bit-congestible or packet-congestible (these terms are defined in Section 1.1).

Notation: To be concrete, we will compare two flows with different packet sizes, s_1 and s_2 . As an example, we will take $s_1 = 60B = 480b$ and $s_2 = 1500B = 12,000b$.

A flow's bit rate, x [bps], is related to its packet rate, u [pps], by

$$x(t) = s.u(t).$$

In the bit-congestible case, path congestion will be denoted by `p_b`, and in the packet-congestible case by `p_p`. When either case is implied, the letter `p` alone will denote path congestion.

B.1. Packet-Size (In)Dependence in Transports

In all cases we consider a packet-mode drop queue that indicates congestion by dropping (or marking) packets with probability `p` irrespective of packet size. We use an example value of loss (marking) probability, `p=0.1%`.

A transport like RFC5681 TCP treats a congestion notification on any packet whatever its size as one event. However, a network with just the packet-mode drop algorithm does give more information if the transport chooses to use it. We will use Table 5 to illustrate this.

We will set aside the last column until later. The columns labelled "Flow 1" and "Flow 2" compare two flows consisting of 60B and 1500B packets respectively. The body of the table considers two separate cases, one where the flows have equal bit-rate and the other with equal packet-rates. In both cases, the two flows fill a 96Mbps link. Therefore, in the equal bit-rate case they each have half the bit-rate (48Mbps). Whereas, with equal packet-rates, flow 1 uses 25 times smaller packets so it gets 25 times less bit-rate--it only gets $1/(1+25)$ of the link capacity ($96\text{Mbps}/26 = 4\text{Mbps}$ after rounding). In contrast flow 2 gets 25 times more bit-rate (92Mbps) in the equal packet rate case because its packets are 25 times larger. The packet rate shown for each flow could easily be derived once the bit-rate was known by dividing bit-rate by packet size, as shown in the column labelled "Formula".

Parameter	Formula	Flow 1	Flow 2	Combined
-----	-----	-----	-----	-----
Packet size	$s/8$	60B	1,500B	(Mix)
Packet size	s	480b	12,000b	(Mix)
Pkt loss probability	p	0.1%	0.1%	0.1%
EQUAL BIT-RATE CASE				
Bit-rate	x	48Mbps	48Mbps	96Mbps
Packet-rate	$u = x/s$	100kpps	4kpps	104kpps
Absolute pkt-loss-rate	$p*u$	100pps	4pps	104pps
Absolute bit-loss-rate	$p*u*s$	48kbps	48kbps	96kbps
Ratio of lost/sent pkts	$p*u/u$	0.1%	0.1%	0.1%
Ratio of lost/sent bits	$p*u*s/(u*s)$	0.1%	0.1%	0.1%
EQUAL PACKET-RATE CASE				
Bit-rate	x	4Mbps	92Mbps	96Mbps
Packet-rate	$u = x/s$	8kpps	8kpps	15kpps
Absolute pkt-loss-rate	$p*u$	8pps	8pps	15pps
Absolute bit-loss-rate	$p*u*s$	4kbps	92kbps	96kbps
Ratio of lost/sent pkts	$p*u/u$	0.1%	0.1%	0.1%
Ratio of lost/sent bits	$p*u*s/(u*s)$	0.1%	0.1%	0.1%

Table 5: Absolute Loss Rates and Loss Ratios for Flows of Small and Large Packets and Both Combined

So far we have merely set up the scenarios. We now consider congestion notification in the scenario. Two TCP flows with the same round trip time aim to equalise their packet-loss-rates over time. That is the number of packets lost in a second, which is the packets per second (u) multiplied by the probability that each one is dropped (p). Thus TCP converges on the "Equal packet-rate" case, where both flows aim for the same "Absolute packet-loss-rate" (both 8pps in the table).

Packet-mode drop actually gives flows sufficient information to measure their loss-rate in bits per second, if they choose, not just packets per second. Each flow can count the size of a lost or marked packet and scale its rate-response in proportion (as TFRC-SP does). The result is shown in the row entitled "Absolute bit-loss-rate", where the bits lost in a second is the packets per second (u) multiplied by the probability of losing a packet (p) multiplied by the packet size (s). Such an algorithm would try to remove any imbalance in bit-loss-rate such as the wide disparity in the "Equal packet-rate" case (4kbps vs. 92kbps). Instead, a packet-size-dependent algorithm would aim for equal bit-loss-rates, which would drive both flows towards the "Equal bit-rate" case, by driving them to equal bit-loss-rates (both 48kbps in this example).

The explanation so far has assumed that each flow consists of packets of only one constant size. Nonetheless, it extends naturally to flows with mixed packet sizes. In the right-most column of Table 5 a flow of mixed size packets is created simply by considering flow 1 and flow 2 as a single aggregated flow. There is no need for a flow to maintain an average packet size. It is only necessary for the transport to scale its response to each congestion indication by the size of each individual lost (or marked) packet. Taking for example the "Equal packet-rate" case, in one second about 8 small packets and 8 large packets are lost (making closer to 15 than 16 losses per second due to rounding). If the transport multiplies each loss by its size, in one second it responds to $8 \times 480\text{b}$ and $8 \times 12,000\text{b}$ lost bits, adding up to 96,000 lost bits in a second. This double checks correctly, being the same as 0.1% of the total bit-rate of 96Mbps. For completeness, the formula for absolute bit-loss-rate is $p(u_1 \times s_1 + u_2 \times s_2)$.

Incidentally, a transport will always measure the loss probability the same irrespective of whether it measures in packets or in bytes. In other words, the ratio of lost to sent packets will be the same as the ratio of lost to sent bytes. (This is why TCP's bit rate is still proportional to packet size even when byte-counting is used, as recommended for TCP in [RFC5681], mainly for orthogonal security reasons.) This is intuitively obvious by comparing two example flows; one with 60B packets, the other with 1500B packets. If both flows pass through a queue with drop probability 0.1%, each flow will lose 1 in 1,000 packets. In the stream of 60B packets the ratio of bytes lost to sent will be 60B in every 60,000B; and in the stream of 1500B packets, the loss ratio will be 1,500B out of 1,500,000B. When the transport responds to the ratio of lost to sent packets, it will measure the same ratio whether it measures in packets or bytes: 0.1% in both cases. The fact that this ratio is the same whether measured in packets or bytes can be seen in Table 5, where the ratio of lost to sent packets and the ratio of lost to sent bytes is always 0.1% in all cases (recall that the scenario was set up with $p=0.1\%$).

This discussion of how the ratio can be measured in packets or bytes is only raised here to highlight that it is irrelevant to this memo! Whether a transport depends on packet size or not depends on how this ratio is used within the congestion control algorithm.

So far we have shown that packet-mode drop passes sufficient information to the transport layer so that the transport can take account of bit-congestion, by using the sizes of the packets that indicate congestion. We have also shown that the transport can choose not to take packet size into account if it wishes. We will now consider whether the transport can know which to do.

B.2. Bit-Congestible and Packet-Congestible Indications

As a thought-experiment, imagine an idealised congestion notification protocol that supports both bit-congestible and packet-congestible resources. It would require at least two ECN flags, one for each of bit-congestible and packet-congestible resources.

1. A packet-congestible resource trying to code congestion level p_p into a packet stream should mark the idealised 'packet congestion' field in each packet with probability p_p irrespective of the packet's size. The transport should then take a packet with the packet congestion field marked to mean just one mark, irrespective of the packet size.
2. A bit-congestible resource trying to code time-varying byte-congestion level p_b into a packet stream should mark the 'byte congestion' field in each packet with probability p_b , again irrespective of the packet's size. Unlike before, the transport should take a packet with the byte congestion field marked to count as a mark on each byte in the packet.

This hides a fundamental problem--much more fundamental than whether we can magically create header space for yet another ECN flag, or whether it would work while being deployed incrementally. Distinguishing drop from delivery naturally provides just one implicit bit of congestion indication information--the packet is either dropped or not. It is hard to drop a packet in two ways that are distinguishable remotely. This is a similar problem to that of distinguishing wireless transmission losses from congestive losses.

This problem would not be solved even if ECN were universally deployed. A congestion notification protocol must survive a transition from low levels of congestion to high. Marking two states is feasible with explicit marking, but much harder if packets are dropped. Also, it will not always be cost-effective to implement AQM at every low level resource, so drop will often have to suffice.

We are not saying two ECN fields will be needed (and we are not saying that somehow a resource should be able to drop a packet in one of two different ways so that the transport can distinguish which sort of drop it was!). These two congestion notification channels are a conceptual device to illustrate a dilemma we could face in the future. Section 3 gives four good reasons why it would be a bad idea to allow for packet size by biasing drop probability in favour of small packets within the network. The impracticality of our thought experiment shows that it will be hard to give transports a practical way to know whether to take account of the size of congestion indication packets or not.

Fortunately, this dilemma is not pressing because by design most equipment becomes bit-congested before its packet-processing becomes congested (as already outlined in Section 1.1). Therefore transports can be designed on the relatively sound assumption that a congestion indication will usually imply bit-congestion.

Nonetheless, although the above idealised protocol isn't intended for implementation, we do want to emphasise that research is needed to predict whether there are good reasons to believe that packet congestion might become more common, and if so, to find a way to somehow distinguish between bit and packet congestion [RFC3714].

Recently, the dual resource queue (DRQ) proposal [DRQ] has been made on the premise that, as network processors become more cost effective, per packet operations will become more complex (irrespective of whether more function in the network is desirable). Consequently the premise is that CPU congestion will become more common. DRQ is a proposed modification to the RED algorithm that folds both bit congestion and packet congestion into one signal (either loss or ECN).

Finally, we note one further complication. Strictly, packet-congestible resources are often cycle-congestible. For instance, for routing look-ups load depends on the complexity of each look-up and whether the pattern of arrivals is amenable to caching or not. This also reminds us that any solution must not require a forwarding engine to use excessive processor cycles in order to decide how to say it has no spare processor cycles.

Appendix C. Byte-mode Drop Complicates Policing Congestion Response

This section is informative, not normative.

There are two main classes of approach to policing congestion response: i) policing at each bottleneck link or ii) policing at the edges of networks. Packet-mode drop in RED is compatible with either, while byte-mode drop precludes edge policing.

The simplicity of an edge policer relies on one dropped or marked packet being equivalent to another of the same size without having to know which link the drop or mark occurred at. However, the byte-mode drop algorithm has to depend on the local MTU of the line--it needs to use some concept of a 'normal' packet size. Therefore, one dropped or marked packet from a byte-mode drop algorithm is not necessarily equivalent to another from a different link. A policing function local to the link can know the local MTU where the congestion occurred. However, a policer at the edge of the network cannot, at least not without a lot of complexity.

The early research proposals for type (i) policing at a bottleneck link [pBox] used byte-mode drop, then detected flows that contributed disproportionately to the number of packets dropped. However, with no extra complexity, later proposals used packet mode drop and looked for flows that contributed a disproportionate amount of dropped bytes [CHOKe_Var_Pkt].

Work is progressing on the congestion exposure protocol (ConEx [RFC6789]), which enables a type (ii) edge policer located at a user's attachment point. The idea is to be able to take an integrated view of the effect of all a user's traffic on any link in the internetwork. However, byte-mode drop would effectively preclude such edge policing because of the MTU issue above.

Indeed, making drop probability depend on the size of the packets that bits happen to be divided into would simply encourage the bits to be divided into smaller packets in order to confuse policing. In contrast, as long as a dropped/marked packet is taken to mean that all the bytes in the packet are dropped/marked, a policer can remain robust against bits being re-divided into different size packets or across different size flows [Rate_fair_Dis].

Appendix D. Changes from Previous Versions

To be removed by the RFC Editor on publication.

Full incremental diffs between each version are available at
<<http://tools.ietf.org/wg/tsvwg/draft-ietf-tsvwg-byte-pkt-congest/>>
(courtesy of the rfcdiff tool):

From -11 to -12: Following the second pass through the IESG:

- * Section 2.1 [Barry Leiba]:
 - + s/No other choice makes sense,/Subject to the exceptions below, no other choice makes sense,/
 - + s/Exceptions to these recommendations MAY be necessary /Exceptions to these recommendations may be necessary /
- * Sections 3.2 and 4.2.3 [Joel Jaeggli]:
 - + Added comment to section 4.2.3 that the examples given are not in widespread production use, but they give evidence that it is possible to follow the advice given.
 - + Section 4.2.3:

- OLD: Although there are no known proposals, it would also be possible and perfectly valid to make control packets robust against drop by explicitly requesting a lower drop probability using their Diffserv code point [RFC2474] to request a scheduling class with lower drop.
NEW: Although there are no known proposals, it would also be possible and perfectly valid to make control packets robust against drop by requesting a scheduling class with lower drop probability, by re-marking to a Diffserv code point [RFC2474] within the same behaviour aggregate.
- appended "Similarly applications, over non-TCP transports could make any packets that are effectively control packets more robust by using Diffserv, data duplication, FEC etc."
- + Updated Wischik ref and added "Reducing Web Latency: the Virtue of Gentle Aggression" ref.
- * Expanded more abbreviations (CoDel, PIE, MTU).
- * Section 1. Intro [Stephen Farrell]:
 - + In the places where the doc describes the dichotomy between 'long-term goal' and 'expediency' the words long term goal and expedient have been introduced, to more explicitly refer back to this introductory para (S.2.1 & S.2.3).
 - + Added explanation of what scaling with packet size means.
- * Conclusions [Benoit Claise]:
 - + OLD: For the specific case of RED, this means that byte-mode queue measurement will often be appropriate although byte-mode drop is strongly deprecated.
NEW: For the specific case of RED, this means that byte-mode queue measurement will often be appropriate but the use of byte-mode drop is very strongly discouraged.

From -10 to -11: Following a further WGLC:

- * Abstract: clarified that advice applies to all AQMs including newer ones
- * Abstract & Intro: changed 'read' to 'detect', because you don't read losses, you detect them.

- * S.1. Introduction: Disambiguated summary of advice on queue measurement.
- * Clarified that the doc deprecates any preference based solely on packet size, it's not only against preferring smaller packets.
- * S.4.1.2. Congestion Measurement without a Queue: Explained that a queue of TXOPs represents a queue into spectrum congested by too many bits.
- * S.5.2: Bit- & Packet-congestible Network: Referred to explanation in S.4.1.2 to make the point that TXOPs are not a primary unit of workload like bits and packets are, even though you get queues of TXOPs.
- * 6. Security: Disambiguated 'bias towards'.
- * 8. Conclusions: Made consistent with recommendation to use time if possible for queue measurement.

From -09 to -10: Following IESG review:

- * Updates 2309: Left header unchanged reflecting eventual IESG consensus [Sean Turner, Pete Resnick].
- * S.1 Intro: This memo adds to the congestion control principles enumerated in BCP 41 [Pete Resnick]
- * Abstract, S.1, S.1.1, s.1.2 Intro, Scoping and Example: Made applicability to all AQMs clearer listing some more example AQMs and explained that we always use RED for examples, but this doesn't mean it's not applicable to other AQMs. [A number of reviewers have described the draft as "about RED"]
- * S.1 & S.2.1 Queue measurement: Explained that the choice between measuring the queue in packets or bytes is only relevant if measuring it in time units is infeasible [So as not to imply that we haven't noticed the advances made by PDPC & CoDel]
- * S.1.1. Terminology: Better explained why hybrid systems congested by both packets and bytes are often designed to be treated as bit-congestible [Richard Barnes].
- * S.2.1. Queue measurement advice: Added examples. Added a counter-example to justify SHOULDs rather than MUSTs. Pointed to S.4.1 for a list of more complicated scenarios. [Benson]

Schliesser, OpsDir]

- * S2.2. Recommendation on Encoding Congestion Notification: Removed SHOULD treat packets equally, leaving only SHOULD NOT drop dependent on packet size, to avoid it sounding like we're saying QoS is not allowed. Pointed to possible app-specific legacy use of byte-mode as a counter-example that prevents us saying MUST NOT. [Pete Resnick]
- * S.2.3. Recommendation on Responding to Congestion: capitalised the two SHOULDs in recommendations for TCP, and gave possible counter-examples. [noticed while dealing with Pete Resnick's point]
- * S2.4. Splitting & Merging: RTCP -> RTP/RTCP [Pete McCann, Gen-ART]
- * S.3.2 Small != Control: many control packets are small -> ...tend to be small [Stephen Farrell]
- * S.3.1 Perverse incentives: Changed transport designers to app developers [Stephen Farrell]
- * S.4.1.1. Fixed Size Packet Buffers: Nearly completely re-written to simplify and to reverse the advice when the underlying resource is bit-congestible, irrespective of whether the buffer consists of fixed-size packet buffers. [Richard Barnes & Benson Schliesser]
- * S.4.2.1.2. Packet Size Bias Regardless of AQM: Largely re-written to reflect the earlier change in advice about fixed-size packet buffers, and to primarily focus on getting rid of tail-drop, not various nuances of tail-drop. [Richard Barnes & Benson Schliesser]
- * Editorial corrections [Tim Bray, AppsDir, Pete McCann, Gen-ART and others]
- * Updated refs (two I-Ds have become RFCs). [Pete McCann]

From -08 to -09: Following WG last call:

- * S.2.1: Made RED-related queue measurement recommendations clearer
- * S.2.3: Added to "Recommendation on Responding to Congestion" to make it clear that we are definitely not saying transports have to equalise bit-rates, just how to do it and not do it, if you

want to.

- * S.3: Clarified motivation sections S.3.3 "Transport-Independent Network" and S.3.5 "Implementation Efficiency"
- * S.3.4: Completely changed motivating argument from "Scaling Congestion Control with Packet Size" to "Partial Deployment of AQM".

From -07 to -08:

- * Altered abstract to say it provides best current practice and highlight that it updates RFC2309
- * Added null IANA section
- * Updated refs

From -06 to -07:

- * A mix-up with the corollaries and their naming in 2.1 to 2.3 fixed.

From -05 to -06:

- * Primarily editorial fixes.

From -04 to -05:

- * Changed from Informational to BCP and highlighted non-normative sections and appendices
- * Removed language about consensus
- * Added "Example Comparing Packet-Mode Drop and Byte-Mode Drop"
- * Arranged "Motivating Arguments" into a more logical order and completely rewrote "Transport-Independent Network" & "Scaling Congestion Control with Packet Size" arguments. Removed "Why Now?"
- * Clarified applicability of certain recommendations
- * Shifted vendor survey to an Appendix
- * Cut down "Outstanding Issues and Next Steps"

- * Re-drafted the start of the conclusions to highlight the three distinct areas of concern
- * Completely re-wrote appendices
- * Editorial corrections throughout.

From -03 to -04:

- * Reordered Sections 2 and 3, and some clarifications here and there based on feedback from Colin Perkins and Mirja Kuehlewind.

From -02 to -03 (this version)

- * Structural changes:
 - + Split off text at end of "Scaling Congestion Control with Packet Size" into new section "Transport-Independent Network"
 - + Shifted "Recommendations" straight after "Motivating Arguments" and added "Conclusions" at end to reinforce Recommendations
 - + Added more internal structure to Recommendations, so that recommendations specific to RED or to TCP are just corollaries of a more general recommendation, rather than being listed as a separate recommendation.
 - + Renamed "State of the Art" as "Critical Survey of Existing Advice" and retitled a number of subsections with more descriptive titles.
 - + Split end of "Congestion Coding: Summary of Status" into a new subsection called "RED Implementation Status".
 - + Removed text that had been in the Appendix "Congestion Notification Definition: Further Justification".
- * Reordered the intro text a little.
- * Made it clearer when advice being reported is deprecated and when it is not.
- * Described AQM as in network equipment, rather than saying "at the network layer" (to side-step controversy over whether functions like AQM are in the transport layer but in network

equipment).

- * Minor improvements to clarity throughout

From -01 to -02:

- * Restructured the whole document for (hopefully) easier reading and clarity. The concrete recommendation, in RFC2119 language, is now in Section 8.

From -00 to -01:

- * Minor clarifications throughout and updated references

From briscoe-byte-pkt-mark-02 to ietf-byte-pkt-congest-00:

- * Added note on relationship to existing RFCs
- * Posed the question of whether packet-congestion could become common and deferred it to the IRTF ICCRG. Added ref to the dual-resource queue (DRQ) proposal.
- * Changed PCN references from the PCN charter & architecture to the PCN marking behaviour draft most likely to imminently become the standards track WG item.

From -01 to -02:

- * Abstract reorganised to align with clearer separation of issue in the memo.
- * Introduction reorganised with motivating arguments removed to new Section 3.
- * Clarified avoiding lock-out of large packets is not the main or only motivation for RED.
- * Mentioned choice of drop or marking explicitly throughout, rather than trying to coin a word to mean either.
- * Generalised the discussion throughout to any packet forwarding function on any network equipment, not just routers.
- * Clarified the last point about why this is a good time to sort out this issue: because it will be hard / impossible to design new transports unless we decide whether the network or the transport is allowing for packet size.

- * Added statement explaining the horizon of the memo is long term, but with short term expediency in mind.
- * Added material on scaling congestion control with packet size (Section 3.4).
- * Separated out issue of normalising TCP's bit rate from issue of preference to control packets (Section 3.2).
- * Divided up Congestion Measurement section for clarity, including new material on fixed size packet buffers and buffer carving (Section 4.1.1 & Section 4.2.1) and on congestion measurement in wireless link technologies without queues (Section 4.1.2).
- * Added section on 'Making Transports Robust against Control Packet Losses' (Section 4.2.3) with existing & new material included.
- * Added tabulated results of vendor survey on byte-mode drop variant of RED (Table 3).

From -00 to -01:

- * Clarified applicability to drop as well as ECN.
- * Highlighted DoS vulnerability.
- * Emphasised that drop-tail suffers from similar problems to byte-mode drop, so only byte-mode drop should be turned off, not RED itself.
- * Clarified the original apparent motivations for recommending byte-mode drop included protecting SYN's and pure ACK's more than equalising the bit rates of TCP's with different segment sizes. Removed some conjectured motivations.
- * Added support for updates to TCP in progress (ackcc & ecn-syn-ack).
- * Updated survey results with newly arrived data.
- * Pulled all recommendations together into the conclusions.
- * Moved some detailed points into two additional appendices and a note.

* Considerable clarifications throughout.

* 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/>

Jukka Manner
Aalto University
Department of Communications and Networking (Comnet)
P.O. Box 13000
FIN-00076 Aalto
Finland

Phone: +358 9 470 22481
EMail: jukka.manner@aalto.fi
URI: <http://www.netlab.tkk.fi/~jmanner/>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 April 2022

R. R. Stewart
Netflix, Inc.
M. Tüxen
I. Rüngeler
Münster Univ. of Appl. Sciences
25 October 2021

Stream Control Transmission Protocol (SCTP) Network Address Translation
Support
draft-ietf-tsvwg-natsupp-23

Abstract

The Stream Control Transmission Protocol (SCTP) provides a reliable communications channel between two end-hosts in many ways similar to the Transmission Control Protocol (TCP). With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT functions for TCP that allows multiple hosts to reside behind a NAT function and yet share a single IPv4 address, even when two hosts (behind a NAT function) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation (NAPT).

This document describes the protocol extensions needed for the SCTP endpoints and the mechanisms for NAT functions necessary to provide similar features of NAPT in the single point and multipoint traversal scenario.

Finally, a YANG module for SCTP NAT is defined.

Status of This Memo

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

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

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

This Internet-Draft will expire on 28 April 2022.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Conventions	5
3. Terminology	5
4. Motivation and Overview	6
4.1. SCTP NAT Traversal Scenarios	6
4.1.1. Single Point Traversal	7
4.1.2. Multipoint Traversal	7
4.2. Limitations of Classical NAPT for SCTP	8
4.3. The SCTP-Specific Variant of NAT	8
5. Data Formats	13
5.1. Modified Chunks	13
5.1.1. Extended ABORT Chunk	13
5.1.2. Extended ERROR Chunk	14
5.2. New Error Causes	14
5.2.1. VTag and Port Number Collision Error Cause	14
5.2.2. Missing State Error Cause	15
5.2.3. Port Number Collision Error Cause	15
5.3. New Parameters	16
5.3.1. Disable Restart Parameter	16
5.3.2. VTags Parameter	17
6. Procedures for SCTP Endpoints and NAT Functions	18
6.1. Association Setup Considerations for Endpoints	19
6.2. Handling of Internal Port Number and Verification Tag Collisions	19
6.2.1. NAT Function Considerations	19
6.2.2. Endpoint Considerations	20
6.3. Handling of Internal Port Number Collisions	20
6.3.1. NAT Function Considerations	20
6.3.2. Endpoint Considerations	21
6.4. Handling of Missing State	21
6.4.1. NAT Function Considerations	22
6.4.2. Endpoint Considerations	22

6.5.	Handling of Fragmented SCTP Packets by NAT Functions . .	24
6.6.	Multi Point Traversal Considerations for Endpoints . . .	24
7.	SCTP NAT YANG Module	24
7.1.	Tree Structure	24
7.2.	YANG Module	25
8.	Various Examples of NAT Traversals	27
8.1.	Single-homed Client to Single-homed Server	28
8.2.	Single-homed Client to Multi-homed Server	30
8.3.	Multihomed Client and Server	32
8.4.	NAT Function Loses Its State	35
8.5.	Peer-to-Peer Communications	37
9.	Socket API Considerations	42
9.1.	Get or Set the NAT Friendliness (SCTP_NAT_FRIENDLY) . . .	43
10.	IANA Considerations	43
10.1.	New Chunk Flags for Two Existing Chunk Types	43
10.2.	Three New Error Causes	45
10.3.	Two New Chunk Parameter Types	46
10.4.	One New URI	46
10.5.	One New YANG Module	46
11.	Security Considerations	46
12.	Normative References	47
13.	Informative References	48
	Acknowledgments	51
	Authors' Addresses	51

1. Introduction

Stream Control Transmission Protocol (SCTP) [RFC4960] provides a reliable communications channel between two end-hosts in many ways similar to TCP [RFC0793]. With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT functions for TCP that allows multiple hosts to reside behind a NAT function using private-use addresses (see [RFC6890]) and yet share a single IPv4 address, even when two hosts (behind a NAT function) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation (NAPT). Please note that this document focuses on the case where the NAT function maps a single or multiple internal addresses to a single external address and vice versa.

To date, specialized code for SCTP has not yet been added to most NAT functions so that only a translation of IP addresses is supported. The end result of this is that only one SCTP-capable host can successfully operate behind such a NAT function and this host can only be single-homed. The only alternative for supporting legacy NAT functions is to use UDP encapsulation as specified in [RFC6951].

The NAT function in the document refers to NAPT functions described in Section 2.2 of [RFC3022], NAT64 [RFC6146], or DS-Lite AFTR [RFC6333].

This document specifies procedures allowing a NAT function to support SCTP by providing similar features to those provided by a NAPT for TCP (see [RFC5382] and [RFC7857]), UDP (see [RFC4787] and [RFC7857]), and ICMP (see [RFC5508] and [RFC7857]). This document also specifies a set of data formats for SCTP packets and a set of SCTP endpoint procedures to support NAT traversal. An SCTP implementation supporting these procedures can assure that in both single-homed and multi-homed cases a NAT function will maintain the appropriate state without the NAT function needing to change port numbers.

It is possible and desirable to make these changes for a number of reasons:

- * It is desirable for SCTP internal end-hosts on multiple platforms to be able to share a NAT function's external IP address in the same way that a TCP session can use a NAT function.
- * If a NAT function does not need to change any data within an SCTP packet, it will reduce the processing burden of NAT'ing SCTP by not needing to execute the CRC32c checksum used by SCTP.
- * Not having to touch the IP payload makes the processing of ICMP messages by NAT functions easier.

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

When considering SCTP-aware NAT it is possible to have multiple levels of support. At each level, the Internal Host, Remote Host, and NAT function does or does not support the procedures described in this document. The following table illustrates the results of the various combinations of support and if communications can occur between two endpoints.

Internal Host	NAT Function	Remote Host	Communication
Support	Support	Support	Yes
Support	Support	No Support	Limited
Support	No Support	Support	None
Support	No Support	No Support	None
No Support	Support	Support	Limited
No Support	Support	No Support	Limited
No Support	No Support	Support	None
No Support	No Support	No Support	None

Table 1: Communication possibilities

From the table it can be seen that no communication can occur when a NAT function does not support SCTP-aware NAT. This assumes that the NAT function does not handle SCTP packets at all and all SCTP packets sent from behind a NAT function are discarded by the NAT function. In some cases, where the NAT function supports SCTP-aware NAT, but one of the two hosts does not support the feature, communication can possibly occur in a limited way. For example, only one host can have a connection when a collision case occurs.

2. Conventions

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

3. Terminology

This document uses the following terms, which are depicted in Figure 1. Familiarity with the terminology used in [RFC4960] and [RFC5061] is assumed.

Internal-Address (Int-Addr)

An internal address that is known to the internal host.

Internal-Port (Int-Port)

The port number that is in use by the host holding the Internal-Address.

Internal-VTag (Int-VTag)

The SCTP Verification Tag (VTag) (see Section 3.1 of [RFC4960]) that the internal host has chosen for an association. The VTag is a unique 32-bit tag that accompanies any incoming SCTP packet for this association to the Internal-Address.

Remote-Address (Rem-Addr)

The address that an internal host is attempting to contact.

Remote-Port (Rem-Port)

The port number used by the host holding the Remote-Address.

Remote-VTag (Rem-VTag)

The Verification Tag (VTag) (see Section 3.1 of [RFC4960]) that the host holding the Remote-Address has chosen for an association. The VTag is a unique 32-bit tag that accompanies any outgoing SCTP packet for this association to the Remote-Address.

External-Address (Ext-Addr)

An external address assigned to the NAT function, that it uses as a source address when sending packets towards a Remote-Address.

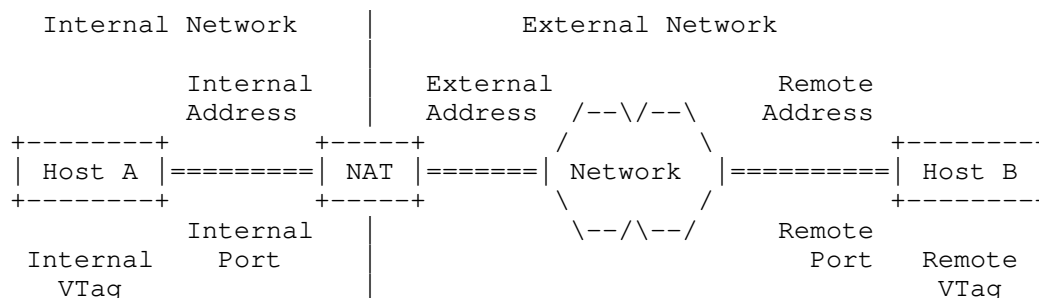


Figure 1: Basic Network Setup

4. Motivation and Overview

4.1. SCTP NAT Traversal Scenarios

This section defines the notion of single and multipoint NAT traversal.

4.1.1. Single Point Traversal

In this case, all packets in the SCTP association go through a single NAT function, as shown in Figure 2.

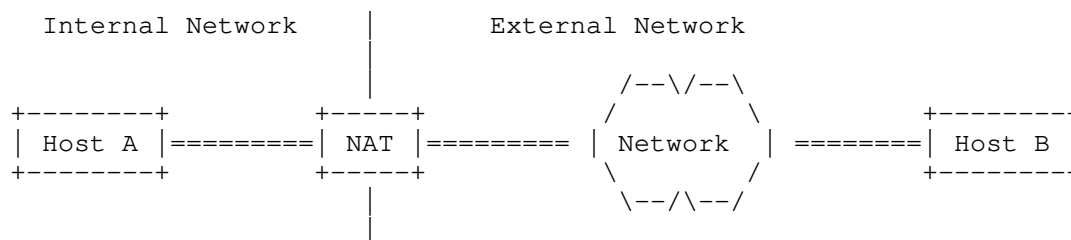


Figure 2: Single NAT Function Scenario

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

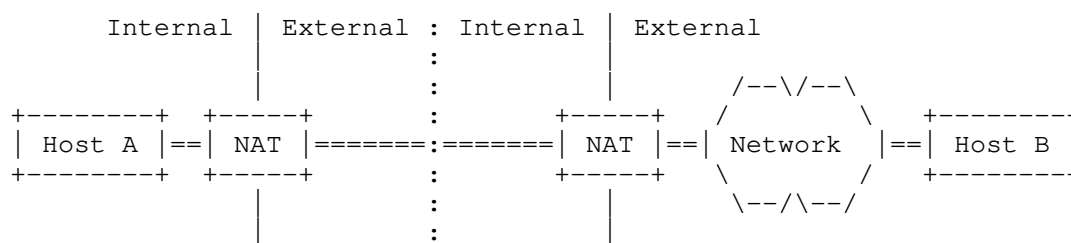


Figure 3: Serial NAT Functions Scenario

Although one of the main benefits of SCTP multi-homing is redundant paths, in the single point traversal scenario the NAT function represents a single point of failure in the path of the SCTP multi-homed association. However, the rest of the path can still benefit from path diversity provided by SCTP multi-homing.

The two SCTP endpoints in this case can be either single-homed or multi-homed. However, the important thing is that the NAT function in this case sees all the packets of the SCTP association.

4.1.2. Multipoint Traversal

This case involves multiple NAT functions and each NAT function only sees some of the packets in the SCTP association. An example is shown in Figure 4.

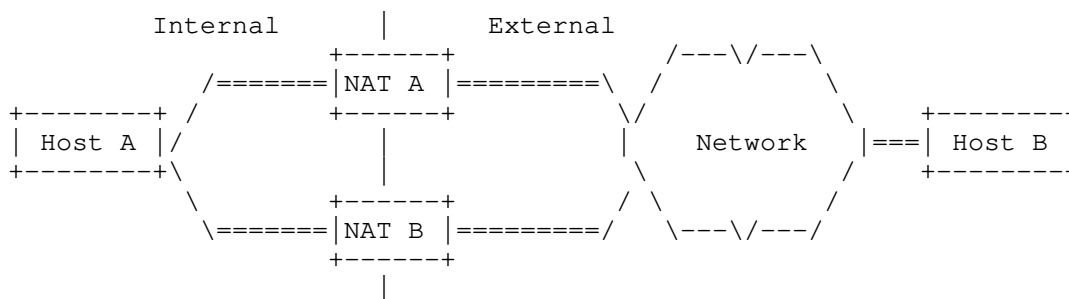


Figure 4: Parallel NAT Functions Scenario

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

4.2. Limitations of Classical NAPT for SCTP

Using classical NAPT possibly results in changing one of the SCTP port numbers during the processing, which requires the recomputation of the transport layer checksum by the NAPT function. Whereas for UDP and TCP this can be done very efficiently, for SCTP the checksum (CRC32c) over the entire packet needs to be recomputed (see Appendix B of [RFC4960] for details of the CRC32c computation). This would considerably add to the NAT computational burden, however hardware support can mitigate this in some implementations.

An SCTP endpoint can have multiple addresses but only has a single port number to use. To make multipoint traversal work, all the NAT functions involved need to recognize the packets they see as belonging to the same SCTP association and perform port number translation in a consistent way. One possible way of doing this is to use a pre-defined table of port numbers and addresses configured within each NAT function. Other mechanisms could make use of NAT to NAT communication. Such mechanisms have not been deployed on a wide scale base and thus are not a preferred solution. Therefore an SCTP variant of NAT function has been developed (see Section 4.3).

4.3. The SCTP-Specific Variant of NAT

In this section it is allowed that there are multiple SCTP capable hosts behind a NAT function that share one External-Address. Furthermore, this section focuses on the single point traversal scenario (see Section 4.1.1).

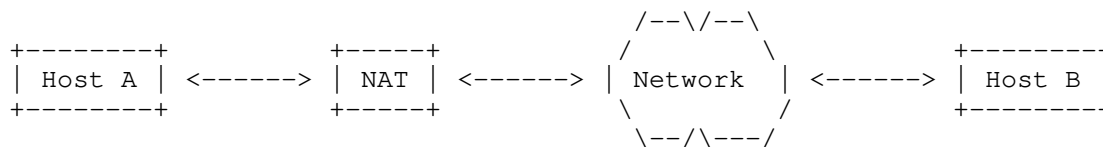
The modification of outgoing SCTP packets sent from an internal host is simple: the source address of the packets has to be replaced with the External-Address. It might also be necessary to establish some state in the NAT function to later handle incoming packets.

Typically, the NAT function has to maintain a NAT binding table of Internal-VTag, Internal-Port, Remote-VTag, Remote-Port, Internal-Address, and whether the restart procedure is disabled or not. An entry in that NAT binding table is called a NAT-State control block. The function Create() obtains the just mentioned parameters and returns a NAT-State control block. A NAT function MAY allow creating NAT-State control blocks via a management interface.

For SCTP packets coming from the external realm of the NAT function the destination address of the packets has to be replaced with the Internal-Address of the host to which the packet has to be delivered, if a NAT state entry is found. The lookup of the Internal-Address is based on the Remote-VTag, Remote-Port, Internal-VTag and the Internal-Port.

The entries in the NAT binding table need to fulfill some uniqueness conditions. There can not be more than one entry NAT binding table with the same pair of Internal-Port and Remote-Port. This rule can be relaxed, if all NAT binding table entries with the same Internal-Port and Remote-Port have the support for the restart procedure disabled (see Section 5.3.1). In this case there can not be no more than one entry with the same Internal-Port, Remote-Port and Remote-VTag and no more than one NAT binding table entry with the same Internal-Port, Remote-Port, and Int-VTag.

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



```

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

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

```

Translate To:

```

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

```

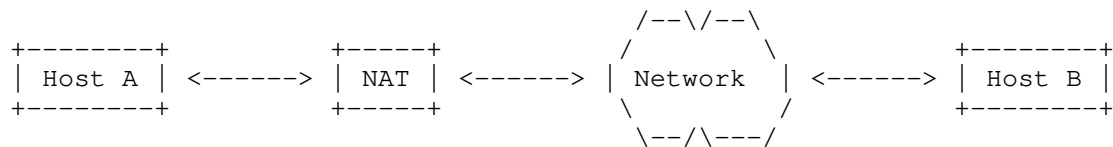
Normally a NAT binding table entry will be created.

However, it is possible that there is already a NAT binding table entry with the same Remote-Port, Internal-Port, and Internal-VTag but different Internal-Address and the restart procedure is disabled. In this case the packet containing the INIT chunk MUST be dropped by the NAT and a packet containing an ABORT chunk SHOULD be sent to the SCTP host that originated the packet with the M bit set and 'VTag and Port Number Collision' error cause (see Section 5.1.1 for the format). The source address of the packet containing the ABORT chunk MUST be the destination address of the packet containing the INIT chunk.

If an outgoing SCTP packet contains an INIT or ASCONF chunk and a matching NAT binding table entry is found, the packet is processed as a normal outgoing packet.

It is also possible that a NAT binding table entry with the same Remote-Port and Internal-Port exists without an Internal-VTag conflict but there exists a NAT binding table entry with the same port numbers but a different Internal-Address and the restart procedure is not disabled. In such a case the packet containing the INIT chunk MUST be dropped by the NAT function and a packet containing an ABORT chunk SHOULD be sent to the SCTP host that originated the packet with the M bit set and 'Port Number Collision' error cause (see Section 5.1.1 for the format).

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

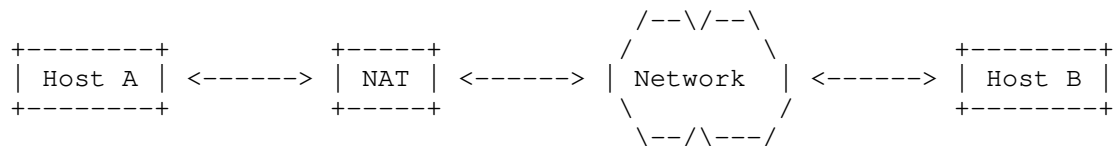


Int-Addr:Int-Port -----> Rem-Addr:Rem-Port
 Rem-VTag

Translate To:

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

The processing of incoming SCTP packets containing an INIT ACK chunk is illustrated in the following figure. The Lookup() function has as input the Internal-VTag, Internal-Port, Remote-VTag, and Remote-Port. It returns the corresponding entry of the NAT binding table and updates the Remote-VTag by substituting it with the value of the Initiate-Tag of the INIT ACK chunk. The wildcard character signifies that the parameter's value is not considered in the Lookup() function or changed in the Update() function, respectively.



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

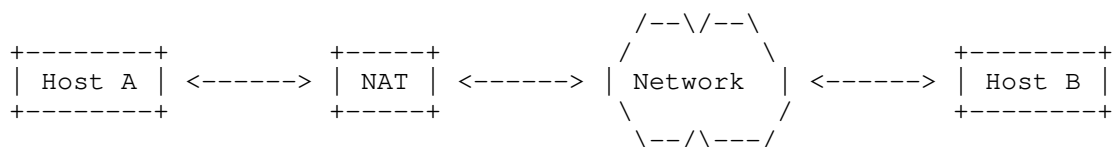
Lookup(Int-VTag, Int-Port, *, Rem-Port)
 Update(*, *, Initiate-Tag, *)

Returns(NAT-State control block containing Int-Addr)

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

In the case where the Lookup function fails because it does not find an entry, the SCTP packet is dropped. If it succeeds, the Update routine inserts the Remote-VTag (the Initiate-Tag of the INIT ACK chunk) in the NAT-State control block.

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



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

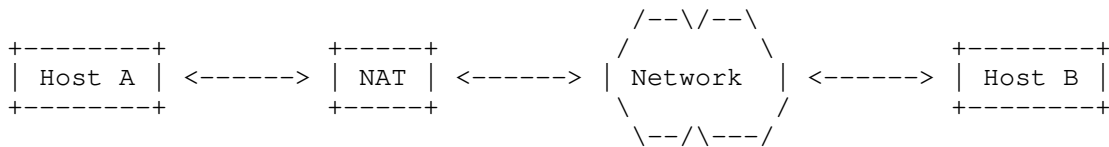
Lookup(*, Int-Port, Rem-VTag, Rem-Port)

Returns (NAT-State control block containing Int-Addr)

Int-Addr:Int-Port <----- Rem-Addr:Rem-Port
Rem-VTag

For an incoming packet containing an INIT chunk a table lookup is made only based on the addresses and port numbers. If an entry with a Remote-VTag of zero is found, it is considered a match and the Remote-VTag is updated. If an entry with a non-matching Remote-VTag is found or no entry is found, the incoming packet is silently dropped. If an entry with a matching Remote-VTag is found, the incoming packet is forwarded. This allows the handling of INIT collision through NAT functions.

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



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

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

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

Int-Addr:Int-Port <----- Rem-Addr:Rem-Port
Int-VTag

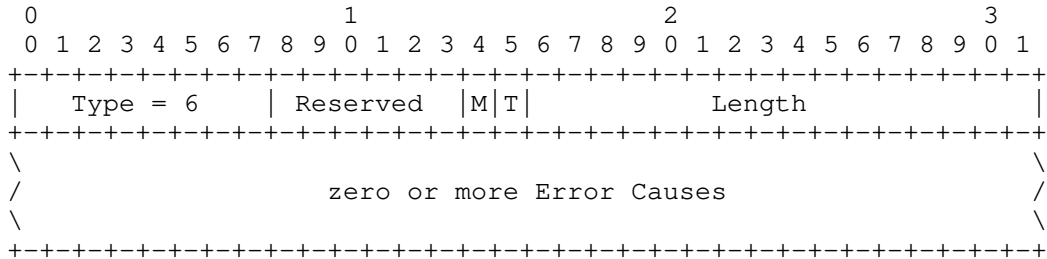
5. Data Formats

This section defines the formats used to support NAT traversal. Section 5.1 and Section 5.2 describe chunks and error causes sent by NAT functions and received by SCTP endpoints. Section 5.3 describes parameters sent by SCTP endpoints and used by NAT functions and SCTP endpoints.

5.1. Modified Chunks

This section presents existing chunks defined in [RFC4960] for which additional flags are specified by this document.

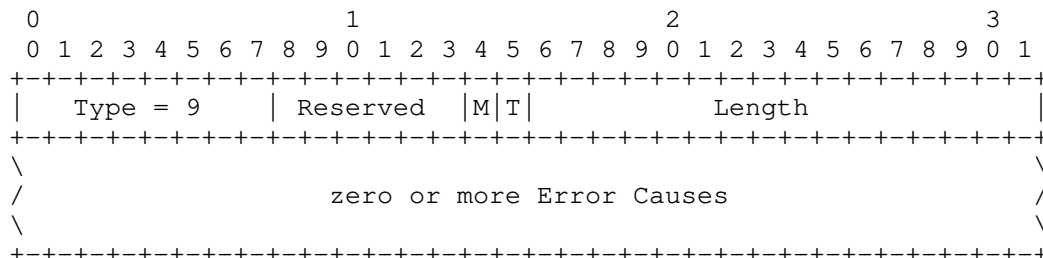
5.1.1. Extended ABORT Chunk



The ABORT chunk is extended to add the new 'M bit'. The M bit indicates to the receiver of the ABORT chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box (e.g., NAT).

[NOTE to RFC-Editor: Assignment of M bit to be confirmed by IANA.]

5.1.2. Extended ERROR Chunk



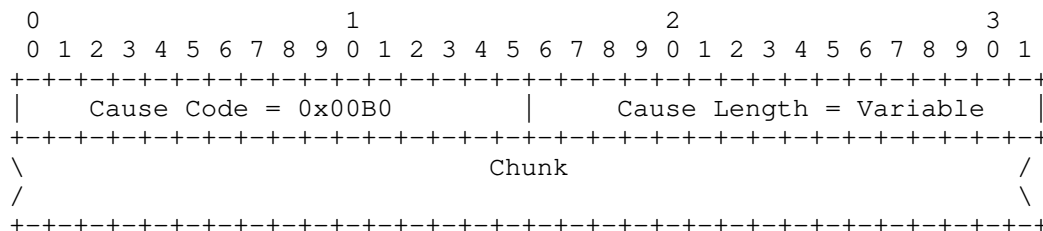
The ERROR chunk defined in [RFC4960] is extended to add the new 'M bit'. The M bit indicates to the receiver of the ERROR chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box.

[NOTE to RFC-Editor: Assignment of M bit to be confirmed by IANA.]

5.2. New Error Causes

This section defines the new error causes added by this document.

5.2.1. VTag and Port Number Collision Error Cause



Cause Code: 2 bytes (unsigned integer)

This field holds the IANA defined cause code for the 'VTag and Port Number Collision' Error Cause. IANA is requested to assign the value 0x00B0 for this cause code.

Cause Length: 2 bytes (unsigned integer)

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

Chunk: variable length

The Cause-Specific Information is filled with the chunk that caused this error. This can be an INIT, INIT ACK, or ASCONF chunk. Note that if the entire chunk will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

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

5.2.2. Missing State Error Cause

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Cause Code = 0x00B1										Cause Length = Variable																													
Original Packet																																							

Cause Code: 2 bytes (unsigned integer)

This field holds the IANA defined cause code for the 'Missing State' Error Cause. IANA is requested to assign the value 0x00B1 for this cause code.

Cause Length: 2 bytes (unsigned integer)

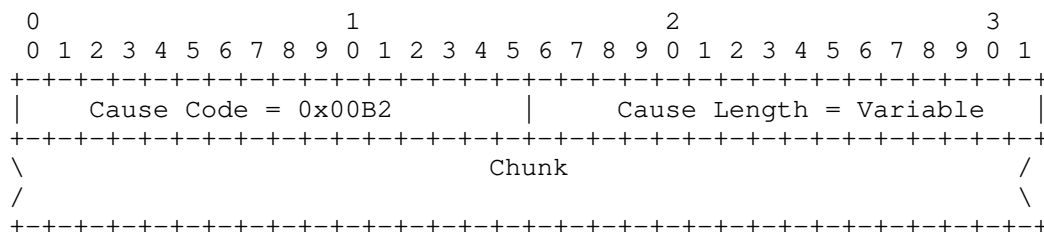
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

Original Packet: variable length

The Cause-Specific Information is filled with the IPv4 or IPv6 packet that caused this error. The IPv4 or IPv6 header MUST be included. Note that if the packet will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

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

5.2.3. Port Number Collision Error Cause



Cause Code: 2 bytes (unsigned integer)

This field holds the IANA defined cause code for the 'Port Number Collision' Error Cause. IANA is requested to assign the value 0x00B2 for this cause code.

Cause Length: 2 bytes (unsigned integer)

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

Chunk: variable length

The Cause-Specific Information is filled with the chunk that caused this error. This can be an INIT, INIT ACK, or ASCONF chunk. Note that if the entire chunk will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

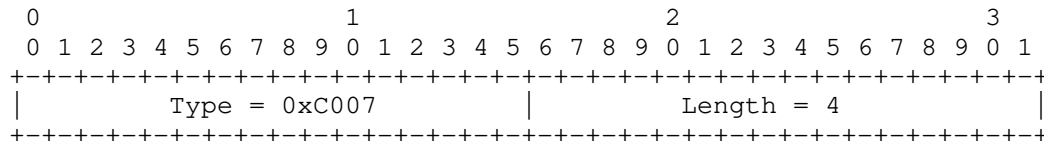
[NOTE to RFC-Editor: Assignment of cause code to be confirmed by IANA.]

5.3. New Parameters

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

5.3.1. Disable Restart Parameter

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



Parameter Type: 2 bytes (unsigned integer)

This field holds the IANA defined parameter type for the Disable Restart Parameter. IANA is requested to assign the value 0xC007 for this parameter type.

Parameter Length: 2 bytes (unsigned integer)

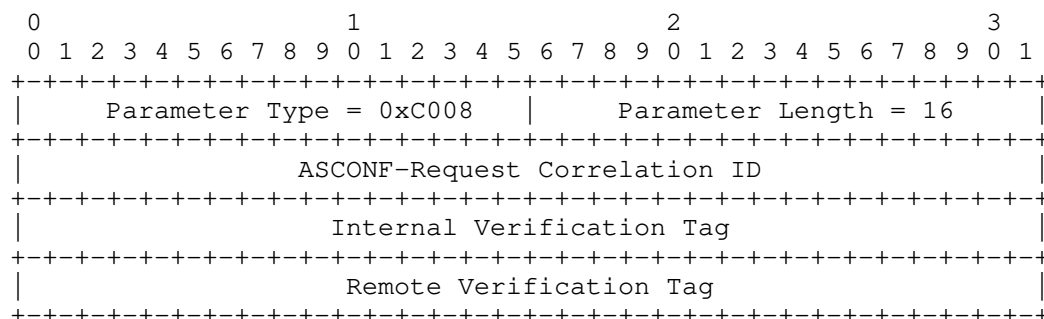
This field holds the length in bytes of the parameter. The value MUST be 4.

[NOTE to RFC-Editor: Assignment of parameter type to be confirmed by IANA.]

The Disable Restart Parameter MAY appear in INIT, INIT ACK and ASCONF chunks and MUST NOT appear in any other chunk.

5.3.2. VTags Parameter

This parameter is used to help a NAT function to recover from state loss.



Parameter Type: 2 bytes (unsigned integer)

This field holds the IANA defined parameter type for the VTags Parameter. IANA is requested to assign the value 0xC008 for this parameter type.

Parameter Length: 2 bytes (unsigned integer)

This field holds the length in bytes of the parameter. The value MUST be 16.

ASCONF-Request Correlation ID: 4 bytes (unsigned integer)

This is an opaque integer assigned by the sender to identify each request parameter. The receiver of the ASCONF Chunk will copy this 32-bit value into the ASCONF Response Correlation ID field of the ASCONF ACK response parameter. The sender of the packet containing the ASCONF chunk can use this same value in the ASCONF ACK chunk to find which request the response is for. The receiver MUST NOT change the value of the ASCONF-Request Correlation ID.

Internal Verification Tag: 4 bytes (unsigned integer)

The Verification Tag that the internal host has chosen for the association. The Verification Tag is a unique 32-bit tag that accompanies any incoming SCTP packet for this association to the Internal-Address.

Remote Verification Tag: 4 bytes (unsigned integer)

The Verification Tag that the host holding the Remote-Address has chosen for the association. The VTag is a unique 32-bit tag that accompanies any outgoing SCTP packet for this association to the Remote-Address.

[NOTE to RFC-Editor: Assignment of parameter type to be confirmed by IANA.]

The VTags Parameter MAY appear in ASCONF chunks and MUST NOT appear in any other chunk.

6. Procedures for SCTP Endpoints and NAT Functions

If an SCTP endpoint is behind an SCTP-aware NAT, a number of problems can arise as it tries to communicate with its peers:

- * IP addresses can not be included in the SCTP packet. This is discussed in Section 6.1.
- * More than one host behind a NAT function could select the same VTag and source port number when communicating with the same peer server. This creates a situation where the NAT function will not be able to tell the two associations apart. This situation is discussed in Section 6.2.
- * If an SCTP endpoint is a server communicating with multiple peers and the peers are behind the same NAT function, then these peers cannot be distinguished by the server. This case is discussed in Section 6.3.
- * A restart of a NAT function during a conversation could cause a loss of its state. This problem and its solution is discussed in Section 6.4.
- * NAT functions need to deal with SCTP packets being fragmented at the IP layer. This is discussed in Section 6.5.
- * An SCTP endpoint can be behind two NAT functions in parallel providing redundancy. The method to set up this scenario is discussed in Section 6.6.

The mechanisms to solve these problems require additional chunks and parameters, defined in this document, and modified handling procedures from those specified in [RFC4960] as described below.

6.1. Association Setup Considerations for Endpoints

The association setup procedure defined in [RFC4960] allows multi-homed SCTP endpoints to exchange its IP-addresses by using IPv4 or IPv6 address parameters in the INIT and INIT ACK chunks. However, this does not work when NAT functions are present.

Every association setup from a host behind a NAT function MUST NOT use multiple internal addresses. The INIT chunk MUST NOT contain an IPv4 Address parameter, IPv6 Address parameter, or Supported Address Types parameter. The INIT ACK chunk MUST NOT contain any IPv4 Address parameter or IPv6 Address parameter using non-global addresses. The INIT chunk and the INIT ACK chunk MUST NOT contain any Host Name parameters.

If the association is intended to be finally multi-homed, the procedure in Section 6.6 MUST be used.

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

6.2. Handling of Internal Port Number and Verification Tag Collisions

Consider the case where two hosts in the Internal-Address space want to set up an SCTP association with the same service provided by some remote hosts. This means that the Remote-Port is the same. If they both choose the same Internal-Port and Internal-VTag, the NAT function cannot distinguish between incoming packets anymore. However, this is unlikely. The Internal-VTags are chosen at random and if the Internal-Ports are also chosen from the ephemeral port range at random (see [RFC6056]) this gives a 46-bit random number that has to match.

The same can happen with the Remote-VTag when a packet containing an INIT ACK chunk or an ASCONF chunk is processed by the NAT function.

6.2.1. NAT Function Considerations

If the NAT function detects a collision of internal port numbers and verification tags, it SHOULD send a packet containing an ABORT chunk with the M bit set if the collision is triggered by a packet containing an INIT or INIT ACK chunk. If such a collision is triggered by a packet containing an ASCONF chunk, it SHOULD send a packet containing an ERROR chunk with the M bit. The M bit is a new

bit defined by this document to express to SCTP that the source of this packet is a "middle" box, not the peer SCTP endpoint (see Section 5.1.1). If a packet containing an INIT ACK chunk triggers the collision, the corresponding packet containing the ABORT chunk MUST contain the same source and destination address and port numbers as the packet containing the INIT ACK chunk. If a packet containing an INIT chunk or an ASCONF chunk, the source and destination address and port numbers MUST be swapped.

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

6.2.2. Endpoint Considerations

The sender of the packet containing the INIT chunk or the receiver of a packet containing the INIT ACK chunk, upon reception of a packet containing an ABORT chunk with M bit set and the appropriate error cause code for colliding NAT binding table state is included, SHOULD reinitiate the association setup procedure after choosing a new initiate tag, if the association is in COOKIE-WAIT state. In any other state, the SCTP endpoint MUST NOT respond.

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

6.3. Handling of Internal Port Number Collisions

When two SCTP hosts are behind an SCTP-aware NAT it is possible that two SCTP hosts in the Internal-Address space will want to set up an SCTP association with the same server running on the same remote host. If the two hosts choose the same internal port, this is considered an internal port number collision.

For the NAT function, appropriate tracking can be performed by assuring that the VTags are unique between the two hosts.

6.3.1. NAT Function Considerations

The NAT function, when processing the packet containing the INIT ACK chunk, SHOULD note in its NAT binding table if the association supports the disable restart extension. This note is used when establishing future associations (i.e. when processing a packet containing an INIT chunk from an internal host) to decide if the connection can be allowed. The NAT function does the following when processing a packet containing an INIT chunk:

- * If the packet containing the INIT chunk is originating from an internal port to a remote port for which the NAT function has no matching NAT binding table entry, it MUST allow the packet containing the INIT chunk creating an NAT binding table entry.
- * If the packet containing the INIT chunk matches an existing NAT binding table entry, it MUST validate that the disable restart feature is supported and, if it does, allow the packet containing the INIT chunk to be forwarded.
- * If the disable restart feature is not supported, the NAT function SHOULD send a packet containing an ABORT chunk with the M bit set.

The 'Port Number Collision' error cause (see Section 5.2.3) MUST be included in the ABORT chunk sent in response to the packet containing an INIT chunk.

If the collision is triggered by a packet containing an ASCONF chunk, a packet containing an ERROR chunk with the 'Port Number Collision' error cause SHOULD be sent in response to the packet containing the ASCONF chunk.

6.3.2. Endpoint Considerations

For the remote SCTP server this means that the Remote-Port and the Remote-Address are the same. If they both have chosen the same Internal-Port the server cannot distinguish between both associations based on the address and port numbers. For the server it looks like the association is being restarted. To overcome this limitation the client sends a Disable Restart parameter in the INIT chunk.

When the server receives this parameter it does the following:

- * It MUST include a Disable Restart parameter in the INIT ACK to inform the client that it will support the feature.
- * It MUST disable the restart procedures defined in [RFC4960] for this association.

Servers that support this feature will need to be capable of maintaining multiple connections to what appears to be the same peer (behind the NAT function) differentiated only by the VTags.

6.4. Handling of Missing State

6.4.1. NAT Function Considerations

If the NAT function receives a packet from the internal network for which the lookup procedure does not find an entry in the NAT binding table, a packet containing an ERROR chunk SHOULD be sent back with the M bit set. The source address of the packet containing the ERROR chunk MUST be the destination address of the packet received from the internal network. The verification tag is reflected and the T bit is set. Such a packet containing an ERROR chunk SHOULD NOT be sent if the received packet contains an ASCONF chunk with the VTags parameter or an ABORT, SHUTDOWN COMPLETE or INIT ACK chunk. A packet containing an ERROR chunk MUST NOT be sent if the received packet contains an ERROR chunk with the M bit set. In any case, the packet SHOULD NOT be forwarded to the remote address.

If the NAT function receives a packet from the internal network for which it has no NAT binding table entry and the packet contains an ASCONF chunk with the VTags parameter, the NAT function MUST update its NAT binding table according to the verification tags in the VTags parameter and, if present, the Disable Restart parameter.

When sending a packet containing an ERROR chunk, the error cause 'Missing State' (see Section 5.2.2) MUST be included and the M bit of the ERROR chunk MUST be set (see Section 5.1.2).

6.4.2. Endpoint Considerations

Upon reception of this packet containing the ERROR chunk by an SCTP endpoint the receiver takes the following actions:

- * It SHOULD validate that the verification tag is reflected by looking at the VTag that would have been included in an outgoing packet. If the validation fails, discard the received packet containing the ERROR chunk.
- * It SHOULD validate that the peer of the SCTP association supports the dynamic address extension. If the validation fails, discard the received packet containing the ERROR chunk.
- * It SHOULD generate a packet containing a new ASCONF chunk containing the VTags parameter (see Section 5.3.2) and the Disable Restart parameter (see Section 5.3.1) if the association is using the disable restart feature. By processing this packet the NAT function can recover the appropriate state. The procedures for generating an ASCONF chunk can be found in [RFC5061].

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

Note that it is possible that upon receiving a packet containing an ASCONF chunk containing the VTags parameter the NAT function will realize that it has an 'Internal Port Number and Verification Tag collision'. In such a case the NAT function SHOULD send a packet containing an ERROR chunk with the error cause code set to 'VTag and Port Number Collision' (see Section 5.2.1).

If an SCTP endpoint receives a packet containing an ERROR chunk with 'Internal Port Number and Verification Tag collision' as the error cause and the packet in the Error Chunk contains an ASCONF with the VTags parameter, careful examination of the association is necessary. The endpoint does the following:

- * It MUST validate that the verification tag is reflected by looking at the VTag that would have been included in the outgoing packet. If the validation fails, it MUST discard the packet.
- * It MUST validate that the peer of the SCTP association supports the dynamic address extension. If the peer does not support this extension, it MUST discard the received packet containing the ERROR chunk.
- * If the association is attempting to add an address (i.e. following the procedures in Section 6.6) then the endpoint MUST NOT consider the address part of the association and SHOULD make no further attempt to add the address (i.e. cancel any ASCONF timers and remove any record of the path), since the NAT function has a VTag collision and the association cannot easily create a new VTag (as it would if the error occurred when sending a packet containing an INIT chunk).
- * If the endpoint has no other path, i.e. the procedure was executed due to missing a state in the NAT function, then the endpoint MUST abort the association. This would occur only if the local NAT function restarted and accepted a new association before attempting to repair the missing state (Note that this is no different than what happens to all TCP connections when a NAT function loses its state).

6.5. Handling of Fragmented SCTP Packets by NAT Functions

SCTP minimizes the use of IP-level fragmentation. However, it can happen that using IP-level fragmentation is needed to continue an SCTP association. For example, if the path MTU is reduced and there are still some DATA chunk in flight, which require packets larger than the new path MTU. If IP-level fragmentation can not be used, the SCTP association will be terminated in a non-graceful way. See [RFC8900] for more information about IP fragmentation.

Therefore, a NAT function MUST be able to handle IP-level fragmented SCTP packets. The fragments MAY arrive in any order.

When an SCTP packet can not be forwarded by the NAT function due to MTU issues and the IP header forbids fragmentation, the NAT MUST send back a "Fragmentation needed and DF set" ICMPv4 or PTB ICMPv6 message to the internal host. This allows for a faster recovery from this packet drop.

6.6. Multi Point Traversal Considerations for Endpoints

If a multi-homed SCTP endpoint behind a NAT function connects to a peer, it MUST first set up the association single-homed with only one address causing the first NAT function to populate its state. Then it SHOULD add each IP address using packets containing ASCONF chunks sent via their respective NAT functions. The address used in the Add IP address parameter is the wildcard address (0.0.0.0 or ::0) and the address parameter in the ASCONF chunk SHOULD also contain the VTags parameter and optionally the Disable Restart parameter.

7. SCTP NAT YANG Module

This section defines a YANG module for SCTP NAT.

The terminology for describing YANG data models is defined in [RFC7950]. The meaning of the symbols in tree diagrams is defined in [RFC8340].

7.1. Tree Structure

This module augments NAT YANG module [RFC8512] with SCTP specifics. The module supports both classical SCTP NAT (that is, rewrite port numbers) and SCTP-specific variant where the ports numbers are not altered. The YANG "feature" is used to indicate whether SCTP-specific variant is supported.

The tree structure of the SCTP NAT YANG module is provided below:

```

module: ietf-nat-sctp
  augment /nat:nat/nat:instances/nat:instance
    /nat:policy/nat:timers:
      +--rw sctp-timeout?  uint32
  augment /nat:nat/nat:instances/nat:instance
    /nat:mapping-table/nat:mapping-entry:
      +--rw int-VTag?      uint32 {sctp-nat}?
      +--rw rem-VTag?      uint32 {sctp-nat}?

```

Concretely, the SCTP NAT YANG module augments the NAT YANG module (policy, in particular) with the following:

- * The sctp-timeout is used to control the SCTP inactivity timeout. That is, the time an SCTP mapping will stay active without SCTP packets traversing the NAT. This timeout can be set only for SCTP. Hence, `"/nat:nat/nat:instances/nat:instance/nat:policy/nat:transport-protocols/nat:protocol-id"` MUST be set to `'132'` (SCTP).

In addition, the SCTP NAT YANG module augments the mapping entry with the following parameters defined in Section 3. These parameters apply only for SCTP NAT mapping entries (i.e., `"/nat/instances/instance/mapping-table/mapping-entry/transport-protocol"` MUST be set to `'132'`);

- * The Internal Verification Tag (Int-VTag)
- * The Remote Verification Tag (Rem-VTag)

7.2. YANG Module

```

<CODE BEGINS> file "ietf-nat-sctp@2020-11-02.yang"
module ietf-nat-sctp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nat-sctp";
  prefix nat-sctp;

  import ietf-nat {
    prefix nat;
    reference
      "RFC 8512: A YANG Module for Network Address Translation
       (NAT) and Network Prefix Translation (NPT)";
  }

  organization
    "IETF TSVWG Working Group";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/tsvwg/>

```

WG List: <mailto:tsvwg@ietf.org>

Author: Mohamed Boucadair
<mailto:mohamed.boucadair@orange.com>;

description

"This module augments NAT YANG module with Stream Control Transmission Protocol (SCTP) specifics. The extension supports both a classical SCTP NAT (that is, rewrite port numbers) and a, SCTP-specific variant where the ports numbers are not altered.

Copyright (c) 2020 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

```
revision 2019-11-18 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: Stream Control Transmission Protocol (SCTP)
      Network Address Translation Support";
}

feature sctp-nat {
  description
    "This feature means that SCTP-specific variant of NAT
      is supported. That is, avoid rewriting port numbers.";
  reference
    "Section 4.3 of RFC XXXX.";
}

augment "/nat:nat/nat:instances/nat:instance"
  + "/nat:policy/nat:timers" {
  when "/nat:nat/nat:instances/nat:instance"
    + "/nat:policy/nat:transport-protocols"
    + "/nat:protocol-id = 132";
  description
    "Extends NAT policy with a timeout for SCTP mapping
      entries.";
```

```
    leaf sctp-timeout {
      type uint32;
      units "seconds";
      description
        "SCTP inactivity timeout. That is, the time an SCTP
        mapping entry will stay active without packets
        traversing the NAT.";
    }
  }

  augment "/nat:nat/nat:instances/nat:instance"
    + "/nat:mapping-table/nat:mapping-entry" {
    when "nat:transport-protocol = 132";
    if-feature "sctp-nat";
    description
      "Extends the mapping entry with SCTP specifics.";

    leaf int-VTag {
      type uint32;
      description
        "The Internal Verification Tag that the internal
        host has chosen for this communication.";
    }
    leaf rem-VTag {
      type uint32;
      description
        "The Remote Verification Tag that the remote
        peer has chosen for this communication.";
    }
  }
}
<CODE ENDS>
```

8. Various Examples of NAT Traversals

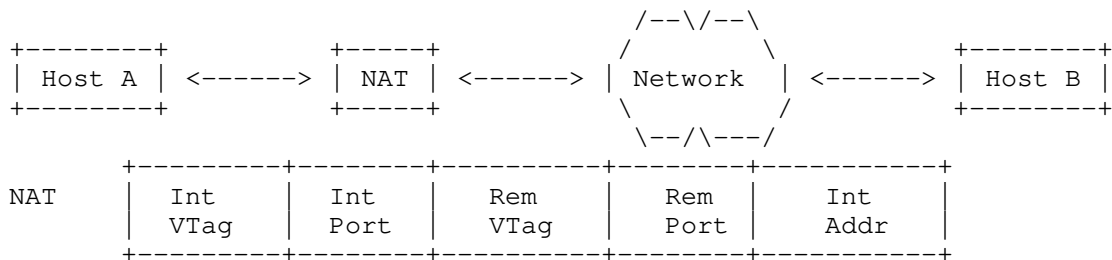
Please note that this section is informational only.

The addresses being used in the following examples are IPv4 addresses for private-use networks and for documentation as specified in [RFC6890]. However, the method described here is not limited to this NAT44 case.

The NAT binding table entries shown in the following examples do not include the flag indicating whether the restart procedure is supported or not. This flag is not relevant for these examples.

8.1. Single-homed Client to Single-homed Server

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



```

INIT[Initiate-Tag = 1234]
10.0.0.1:1 -----> 203.0.113.1:2
    Rem-VTtag = 0

```

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

NAT function creates entry:

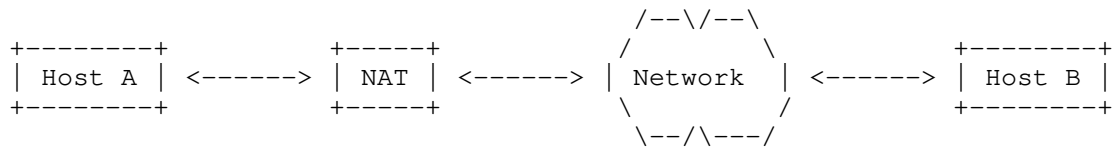
	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
NAT					
	1234	1	0	2	10.0.0.1

```

INIT[Initiate-Tag = 1234]
192.0.2.1:1 -----> 203.0.113.1:2
    Rem-VTtag = 0

```

Host B receives the packet containing an INIT chunk and sends a packet containing an INIT ACK chunk with the NAT's Remote-address as destination address.



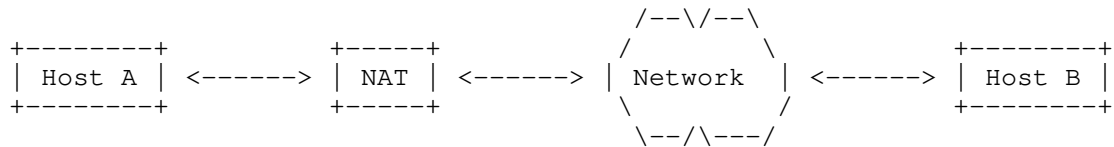
INIT ACK[Initiate-Tag = 5678]
 192.0.2.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

NAT function updates entry:

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

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

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



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

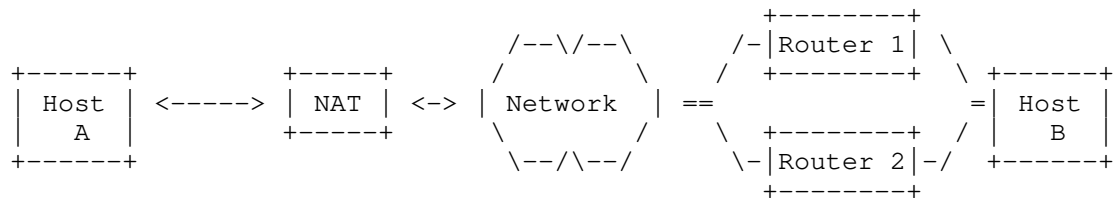
COOKIE ECHO
 192.0.2.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

COOKIE ACK
 192.0.2.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

COOKIE ACK
 10.0.0.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

8.2. Single-homed Client to Multi-homed Server

The internal client is single-homed whereas the remote server is multi-homed. The client (Host A) sends a packet containing an INIT chunk like in the single-homed case.



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

```

INIT[Initiate-Tag = 1234]
10.0.0.1:1 ---> 203.0.113.1:2
Rem-VTag = 0
  
```

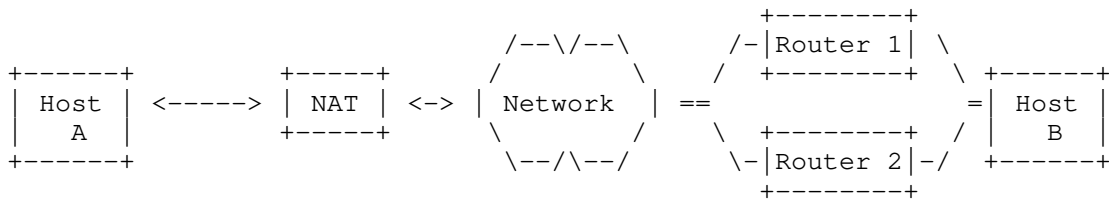
NAT function creates entry:

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

```

INIT[Initiate-Tag = 1234]
192.0.2.1:1 -----> 203.0.113.1:2
Rem-VTag = 0
  
```

The server (Host B) includes its two addresses in the INIT ACK chunk.



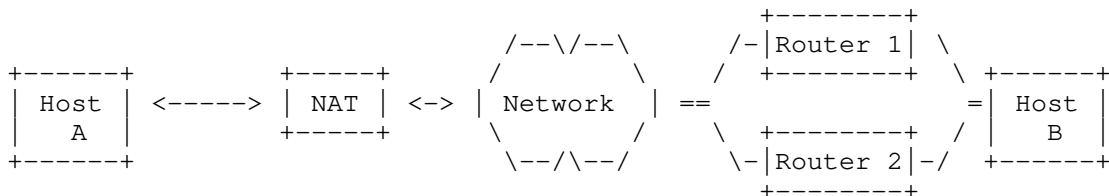
```
INIT ACK[Initiate-tag = 5678, IP-Addr = 203.0.113.129]
192.0.2.1:1 <----- 203.0.113.1:2
Int-VTag = 1234
```

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

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

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

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



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

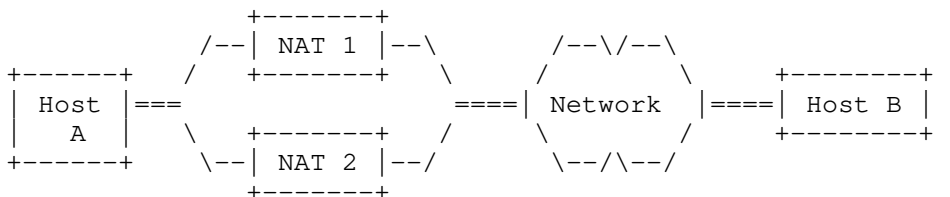
COOKIE ECHO
192.0.2.1:1 -----> 203.0.113.1:2
Rem-VTag = 5678

COOKIE ACK
192.0.2.1:1 <----- 203.0.113.1:2
Int-VTag = 1234

COOKIE ACK
10.0.0.1:1 <--- 203.0.113.1:2
Int-VTag = 1234

8.3. Multihomed Client and Server

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



NAT 1					
	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr

INIT[Initiate-Tag = 1234]
10.0.0.1:1 -----> 203.0.113.1:2
Rem-VTag = 0

NAT function 1 creates entry:

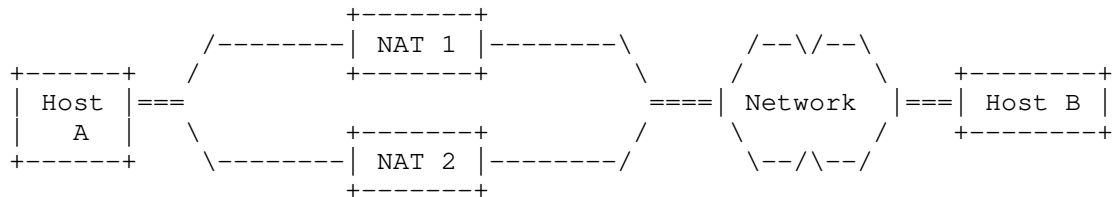
NAT 1	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	0	2	10.0.0.1

```

                                INIT[Initiate-Tag = 1234]
192.0.2.1:1 -----> 203.0.113.1:2
                                Rem-VTag = 0

```

Host B includes its second address in the INIT ACK.



```

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

```

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

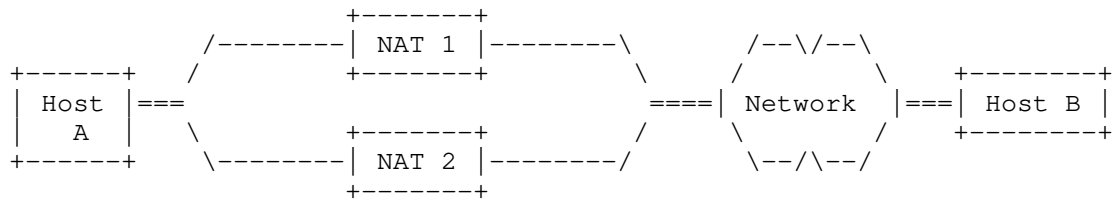
NAT 1	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	5678	2	10.0.0.1

```

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

```

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



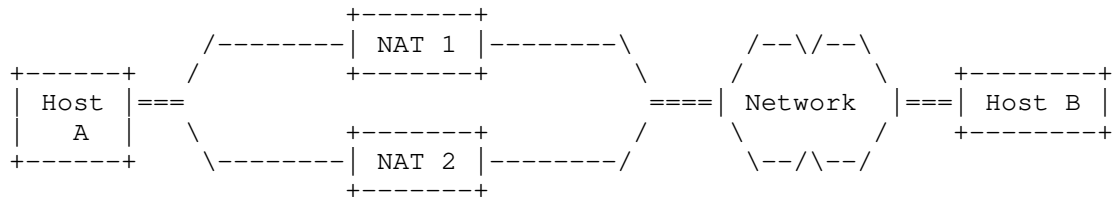
COOKIE ECHO
 10.0.0.1:1 -----> 203.0.113.1:2
 Rem-VTag = 5678

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

COOKIE ACK
 192.0.2.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

COOKIE ACK
 10.0.0.1:1 <----- 203.0.113.1:2
 Int-VTag = 1234

Host A announces its second address in an ASCONF chunk. The address parameter contains a wildcard address (0.0.0.0 or ::0) to indicate that the source address has to be added. The address parameter within the ASCONF chunk will also contain the pair of VTags (remote and internal) so that the NAT function can populate its NAT binding table entry completely with this single packet.



ASCONF [ADD-IP=0.0.0.0, INT-VTag=1234, Rem-VTag = 5678]
 10.1.0.1:1 -----> 203.0.113.129:2
 Rem-VTag = 5678

NAT function 2 creates a complete entry:

NAT 2	+-----+				
	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	1234	1	5678	2	10.1.0.1
	+-----+				

```

ASCONF [ADD-IP, Int-VTag=1234, Rem-VTag = 5678]
192.0.2.129:1 -----> 203.0.113.129:2
                        Rem-VTag = 5678

```

```

                        ASCONF ACK
192.0.2.129:1 <----- 203.0.113.129:2
                        Int-VTag = 1234

```

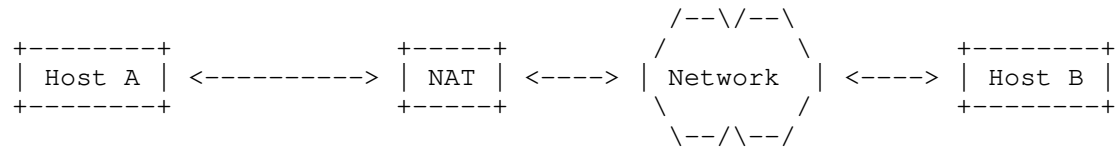
```

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

```

8.4. NAT Function Loses Its State

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



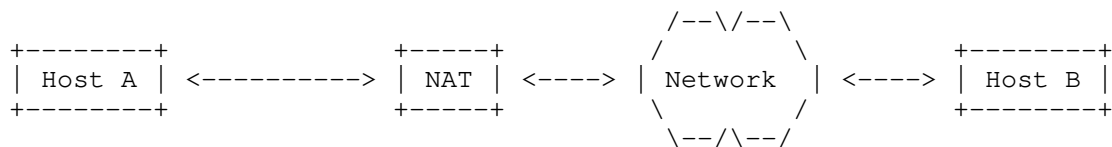
NAT	+-----+				
	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	+-----+				

```

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

```

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

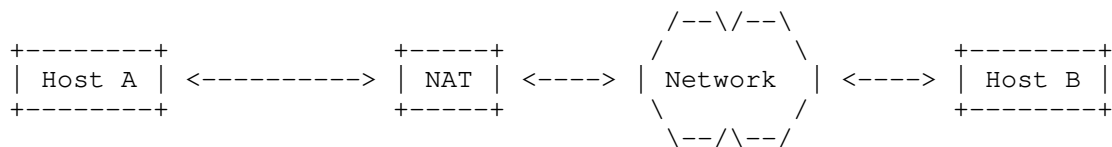


```

ERROR [M bit, NAT state missing]
10.0.0.1:1 <----- 203.0.113.1:2
Rem-VTag = 5678

```

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



```

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

```

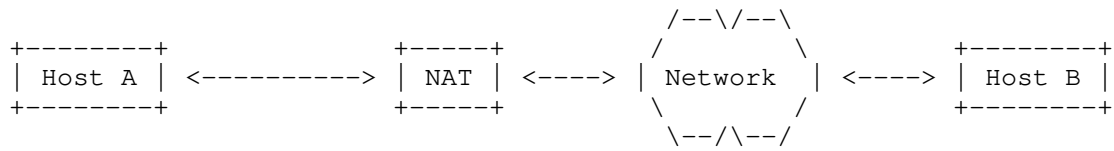
NAT	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
	1234	1	5678	2	10.0.0.1

```

ASCONF [ADD-IP, DELETE-IP, Int-VTag=1234, Rem-VTag = 5678]
192.0.2.2:1 -----> 203.0.113.129:2
Rem-VTag = 5678

```

Host B adds the new source address to this association and deletes all other addresses from this association.



ASCONF ACK
 192.0.2.2:1 <----- 203.0.113.129:2
 Int-VTag = 1234

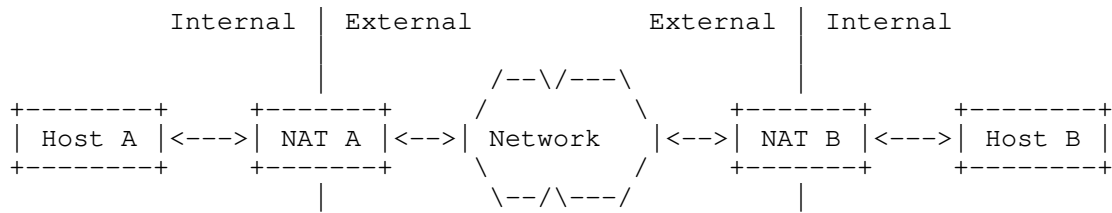
ASCONF ACK
 10.1.0.1:1 <----- 203.0.113.129:2
 Int-VTag = 1234

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

DATA
 192.0.2.2:1 -----> 203.0.113.129:2
 Rem-VTag = 5678

8.5. Peer-to-Peer Communications

If two hosts, each of them behind a NAT function, want to communicate with each other, they have to get knowledge of the peer's external address. This can be achieved with a so-called rendezvous server. Afterwards the destination addresses are external, and the association is set up with the help of the INIT collision. The NAT functions create their entries according to their internal peer's point of view. Therefore, NAT function A's Internal-VTag and Internal-Port are NAT function B's Remote-VTag and Remote-Port, respectively. The naming (internal/remote) of the verification tag in the packet flow is done from the sending host's point of view.



NAT Binding Tables

NAT A	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
-------	-------------	-------------	-------------	-------------	-------------

NAT B	Int v-tag	Int port	Rem v-tag	Rem port	Int Addr
-------	--------------	-------------	--------------	-------------	-------------

```

INIT[Initiate-Tag = 1234]
10.0.0.1:1 --> 203.0.113.1:2
    Rem-VTag = 0
  
```

NAT function A creates entry:

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

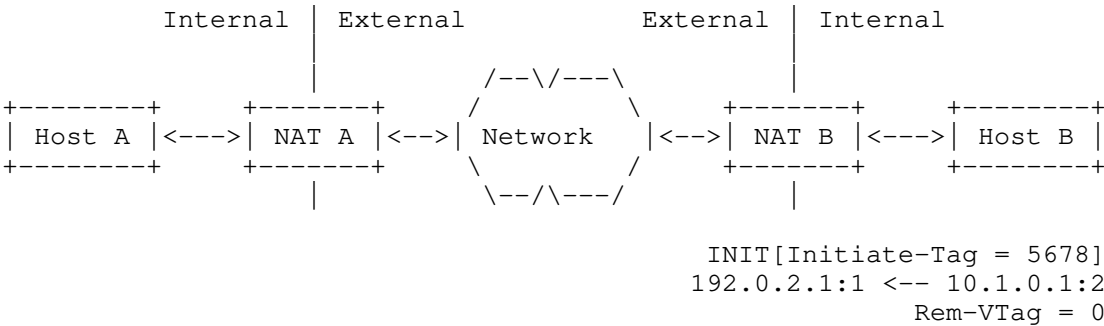
```

INIT[Initiate-Tag = 1234]
192.0.2.1:1 -----> 203.0.113.1:2
    Rem-VTag = 0
  
```

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

NAT B	Int VTag	Int Port	Rem VTag	Rem Port	Int Addr
-------	-------------	-------------	-------------	-------------	-------------

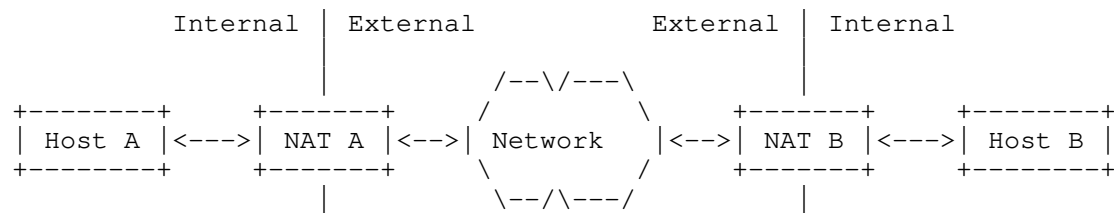
Now Host B sends a packet containing an INIT chunk, which is processed by NAT function B. Its parameters are used to create an entry.



NAT B	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	5678	2	0	1	10.1.0.1

INIT[Initiate-Tag = 5678]
192.0.2.1:1 <----- 203.0.113.1:2
Rem-VTag = 0

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

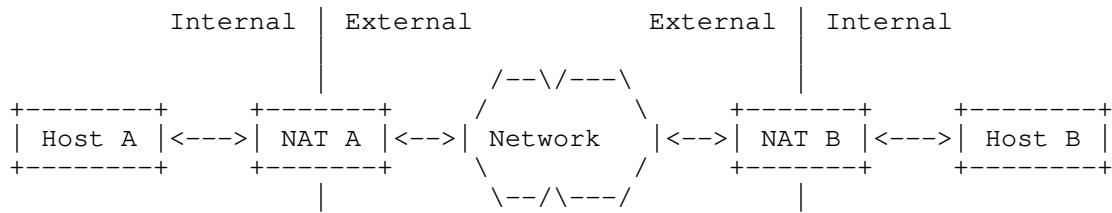


VTag != Int-VTag, but Rem-VTag == 0, find entry.

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

```
INIT[Initiate-tag = 5678]
10.0.0.1:1 <-- 203.0.113.1:2
    Rem-VTag = 0
```

Host A sends a packet containing an INIT ACK chunk, which can pass through NAT function B:



```

INIT ACK[Initiate-Tag = 1234]
10.0.0.1:1 --> 203.0.113.1:2
    Rem-VTag = 5678

```

```

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

```

NAT function B updates entry:

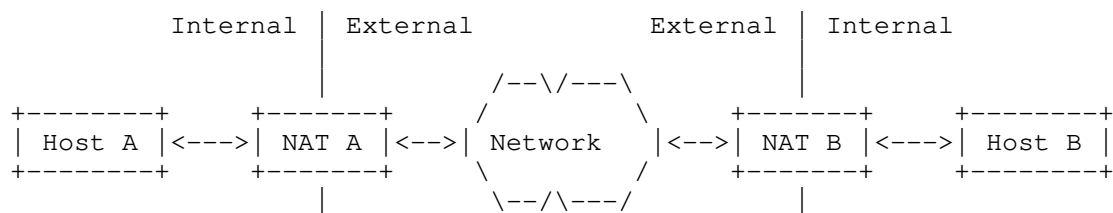
NAT B	Int	Int	Rem	Rem	Int
	VTag	Port	VTag	Port	Addr
	5678	2	1234	1	10.1.0.1

```

INIT ACK[Initiate-Tag = 1234]
192.0.2.1:1 --> 10.1.0.1:2
    Rem-VTag = 5678

```

The lookup for COOKIE ECHO and COOKIE ACK is successful.



COOKIE ECHO
 192.0.2.1:1 <-- 10.1.0.1:2
 Rem-VTag = 1234

COOKIE ECHO
 192.0.2.1:1 <----- 203.0.113.1:2
 Rem-VTag = 1234

COOKIE ECHO
 10.0.0.1:1 <-- 203.0.113.1:2
 Rem-VTag = 1234

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

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

COOKIE ACK
 192.0.2.1:1 --> 10.1.0.1:2
 Rem-VTag = 5678

9. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to control NAT friendliness.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is extended by supporting one new read/write socket option.

9.1. Get or Set the NAT Friendliness (SCTP_NAT_FRIENDLY)

This socket option uses the option_level IPPROTO_SCTP and the option_name SCTP_NAT_FRIENDLY. It can be used to enable/disable the NAT friendliness for future associations and retrieve the value for future and specific ones.

```
struct sctp_assoc_value {  
    sctp_assoc_t assoc_id;  
    uint32_t assoc_value;  
};
```

assoc_id

This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application can fill in an association identifier or SCTP_FUTURE_ASSOC for this query. It is an error to use SCTP_{CURRENT|ALL}_ASSOC in assoc_id.

assoc_value

A non-zero value indicates a NAT-friendly mode.

10. IANA Considerations

[NOTE to RFC-Editor: "RFCXXXX" is to be replaced by the RFC number you assign this document.]

[NOTE to RFC-Editor: The requested values for the chunk type and the chunk parameter types are tentative and to be confirmed by IANA.]

This document (RFCXXXX) is the reference for all registrations described in this section. The requested changes are described below.

10.1. New Chunk Flags for Two Existing Chunk Types

As defined in [RFC6096] two chunk flags have to be assigned by IANA for the ERROR chunk. The requested value for the T bit is 0x01 and for the M bit is 0x02.

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

ERROR Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	T bit	[RFCXXXX]
0x02	M bit	[RFCXXXX]
0x04	Unassigned	
0x08	Unassigned	
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

Table 2

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

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

ABORT Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	T bit	[RFC4960]
0x02	M bit	[RFCXXXX]
0x04	Unassigned	
0x08	Unassigned	
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

Table 3

10.2. Three New Error Causes

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

This requires three additional lines in the "Error Cause Codes" registry for SCTP:

Error Cause Codes

Value	Cause Code	Reference
176	VTag and Port Number Collision	[RFCXXXX]
177	Missing State	[RFCXXXX]
178	Port Number Collision	[RFCXXXX]

Table 4

10.3. Two New Chunk Parameter Types

Two chunk parameter types have to be assigned by IANA. IANA is requested to assign these values from the pool of parameters with the upper two bits set to '11' and to use the values given below.

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

Chunk Parameter Types

ID Value	Chunk Parameter Type	Reference
49159	Disable Restart (0xC007)	[RFCXXXX]
49160	VTags (0xC008)	[RFCXXXX]

Table 5

10.4. One New URI

An URI in the "ns" subregistry within the "IETF XML" registry has to be assigned by IANA ([RFC3688]):

URI: urn:ietf:params:xml:ns:yang:ietf-nat-sctp
 Registrant Contact: The IESG.
 XML: N/A; the requested URI is an XML namespace.

10.5. One New YANG Module

An YANG module in the "YANG Module Names" subregistry within the "YANG Parameters" registry has to be assigned by IANA ([RFC6020]):

Name: ietf-nat-sctp
 Namespace: urn:ietf:params:xml:ns:yang:ietf-nat-sctp
 Maintained by IANA: N
 Prefix: nat-sctp
 Reference: RFCXXXX

11. Security Considerations

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

Generic issues related to address sharing are discussed in [RFC6269] and apply to SCTP as well.

For SCTP endpoints not disabling the restart procedure, this document does not add any additional security considerations to the ones given in [RFC4960], [RFC4895], and [RFC5061].

SCTP endpoints disabling the restart procedure, need to monitor the status of all associations to mitigate resource exhaustion attacks by establishing a lot of associations sharing the same IP addresses and port numbers.

In any case, SCTP is protected by the verification tags and the usage of [RFC4895] against off-path attackers.

For IP-level fragmentation and reassembly related issues see [RFC4963].

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All data nodes defined in the YANG module that can be created, modified, and deleted (i.e., config true, which is the default) are considered sensitive. Write operations (e.g., edit-config) applied to these data nodes without proper protection can negatively affect network operations. An attacker who is able to access the SCTP NAT function can undertake various attacks, such as:

- * Setting a low timeout for SCTP mapping entries to cause failures to deliver incoming SCTP packets.
- * Instantiating mapping entries to cause NAT collision.

12. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, <<https://www.rfc-editor.org/info/rfc4895>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<https://www.rfc-editor.org/info/rfc4960>>.
- [RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <<https://www.rfc-editor.org/info/rfc5061>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, DOI 10.17487/RFC6096, January 2011, <<https://www.rfc-editor.org/info/rfc6096>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <<https://www.rfc-editor.org/info/rfc8512>>.

13. Informative References

- [DOI_10.1145_1496091.1496095]
Hayes, D., But, J., and G. Armitage, "Issues with network address translation for SCTP", ACM SIGCOMM Computer Communication Review Vol. 39, pp. 23-33, DOI 10.1145/1496091.1496095, December 2008, <<https://doi.org/10.1145/1496091.1496095>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <<https://www.rfc-editor.org/info/rfc3022>>.
- [RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <<https://www.rfc-editor.org/info/rfc4787>>.
- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <<https://www.rfc-editor.org/info/rfc4963>>.
- [RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008, <<https://www.rfc-editor.org/info/rfc5382>>.
- [RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, DOI 10.17487/RFC5508, April 2009, <<https://www.rfc-editor.org/info/rfc5508>>.
- [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <<https://www.rfc-editor.org/info/rfc6056>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <<https://www.rfc-editor.org/info/rfc6269>>.
- [RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <<https://www.rfc-editor.org/info/rfc6333>>.
- [RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <<https://www.rfc-editor.org/info/rfc6458>>.
- [RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <<https://www.rfc-editor.org/info/rfc6890>>.
- [RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <<https://www.rfc-editor.org/info/rfc6951>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar, S., and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements", BCP 127, RFC 7857, DOI 10.17487/RFC7857, April 2016, <<https://www.rfc-editor.org/info/rfc7857>>.

- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <<https://www.rfc-editor.org/info/rfc8900>>.

Acknowledgments

The authors wish to thank Mohamed Boucadair, Gorrry Fairhurst, Bryan Ford, David Hayes, Alfred Hines, Karen E. E. Nielsen, Henning Peters, Maksim Proshin, Timo Völker, Dan Wing, and Qiaobing Xie for their invaluable comments.

In addition, the authors wish to thank David Hayes, Jason But, and Grenville Armitage, the authors of [DOI_10.1145_1496091.1496095], for their suggestions.

The authors also wish to thank Mohamed Boucadair for contributing the text related to the YANG module.

Authors' Addresses

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

Email: randall@lakerest.net

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

Email: tuexen@fh-muenster.de

Irene Rüngeler
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: i.ruengeler@fh-muenster.de

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

J. Touch
USC/ISI
April 24, 2015

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

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire on October 24, 2015.

Copyright Notice

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

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

Abstract

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

Table of Contents

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

1. Introduction

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

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

2. Conventions used in this document

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

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

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

3. History

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

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

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

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

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

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

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

4. Current Port Number Use

RFC6335 indicates three ranges of port number assignments:

Binary	Hex	

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

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

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

5. What is a Port Number?

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

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

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

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

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

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

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

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

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

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

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

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

6. Conservation

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

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

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

6.1. Guiding Principles

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

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

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

6.2. Firewall and NAT Considerations

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

Using Dynamic port numbers, or explicitly-indicated port numbers indicated in-band over another service (such as with FTP) often complicates firewall and NAT interactions [RFC959]. FTP over firewalls often requires direct support for deep-packet inspection

(to snoop for the Dynamic port number for the NAT to correctly map) or passive-mode FTP (in which both connections are opened from the client side).

7. Considerations for Requesting Port Number Assignments

Port numbers are assigned by IANA by a set of documented procedures [RFC6335]. The following section describes the steps users can take to help assist with responsible use of assigned port numbers, and with preparing an application for a port number assignment.

7.1. Is a port number assignment necessary?

First, it is useful to consider whether a port number assignment is required. In many cases, a new number assignment may not be needed, for example:

- o Is this really a new service, or can an existing service suffice?
- o Is this an experimental service [RFC3692]? If so, consider using the current experimental ports [RFC2780].
- o Is this service independently useful? Some systems are composed from collections of different service capabilities, but not all component functions are useful as independent services. Port numbers are typically shared among the smallest independently-useful set of functions. Different service uses or properties can be supported in separate pairwise endpoint associations after an initial negotiation, e.g., to support software decomposition.
- o Can this service use a Dynamic port number that is coordinated out-of-band, e.g.:
 - o By explicit configuration of both endpoints.
 - o By internal mechanisms within the same host (e.g., a configuration file, indicated within a URI, or using interprocess communication).
- o Using information exchanged on a related service: FTP, SIP, etc. [RFC959] [RFC3261].
- o Using an existing port discovery service: portmapper, mDNS, etc. [RFC1833] [RFC6762] [RFC6763].

There are a few good examples of reasons that more directly suggest that not only is a port number assignment not necessary, but it is directly counter-indicated:

- o Assigned port numbers are not intended to differentiate performance variations within the same service, e.g., high-speed vs. ordinary speed. Performance variations can be supported within a single assigned port number in context of separate pairwise endpoint associations.
- o Additional assigned port numbers are not intended to replicate an existing service. For example, if a device is configured to use a typical web browser then it the port number used for that service is a copy of the http service that is already assigned to port number 80 and does not warrant a new assignment. However, an automated system that happens to use HTTP framing - but is not primarily accessed by a browser - might be a new service. A good way to tell is "can an unmodified client of the existing service interact with the proposed service"? If so, that service would be a copy of an existing service and would not merit a new assignment.
- o Assigned port numbers not intended for intra-machine communication. Such communication can already be supported by internal mechanisms (interprocess communication, shared memory, shared files, etc.). When Internet communication within a host is desired, the server can bind to a Dynamic port that is indicated to the client using these internal mechanisms.
- o Separate assigned port numbers are not intended for insecure versions of existing (or new) secure services. A service that already requires security would be made more vulnerable by having the same capability accessible without security.

Note that the converse is different, i.e., it can be useful to create a new, secure service that replicates an existing insecure service on a new port number assignment. This can be necessary when the existing service is not backward-compatible with security enhancements, such as the use of TLS [RFC5246] or DTLS [RFC6347].

- o Assigned port numbers are not intended for indicating different service versions. Version differentiation should be handled in-band, e.g., using a version number at the beginning of an association (e.g., connection or other transaction). This may not be possible with legacy assignments, but all new services should incorporate support for version indication.

Some services may not need assigned port numbers at all, e.g., SIP allows voice calls to use Dynamic ports [RFC3261]. Some systems can register services in the DNS, using SRV entries. These services can be discovered by a variety of means, including mDNS, or via direct query [RFC6762] [RFC6763]. In such cases, users can more easily request a SRV name, which are assigned first-come, first-served from a much larger namespace.

IANA assigns port numbers, but this assignment is typically used only for servers, i.e., the host that listens for incoming connections or other associations. Clients, i.e., hosts that initiate connections or other associations, typically refer to those assigned port numbers but do not need port number assignments for their endpoint.

Finally, an assigned port number is not a guarantee of exclusive use. Traffic for any service might appear on any port number, due to misconfiguration or deliberate misuse. Application and service designers are encouraged to validate traffic based on its content.

7.2. How Many Assigned Port Numbers?

As noted earlier, systems might require a single port number assignment, but rarely require multiple port numbers. There are a variety of known ways to reduce assigned port number consumption. Although some may be cumbersome or inefficient, they are nearly always preferable to consuming additional port number assignments.

Such techniques include:

- o Use of a discovery service, either a shared service (mDNS), or a discovery service for a given system [RFC6762] [RFC6763].
- o Multiplex packet types using in-band information, either on a per-message or per-connection basis. Such demultiplexing can even hand-off different messages and connections among different processes, such as is done with FTP [RFC959].

There are some cases where NAT and firewall traversal are significantly improved by having an assigned port number. Although

NAT traversal protocols supporting automatic configuration have been proposed and developed (e.g., STUN [RFC5389], TURN [RFC5766], and ICE [RFC5245]), not all application and service designers can rely on their presence as of yet.

In the past, some services were assigned multiple port numbers or sometimes fairly large port ranges (e.g., X11). This occurred for a variety of reasons: port number conservation was not as widely appreciated, assignments were not as ardently reviewed, etc. This no longer reflects current practice and such assignments are not considered to constitute a precedent for future assignments.

7.3. Picking an Assigned Port Number

Given a demonstrated need for a port number assignment, the next question is how to pick the desired port number. An application for a port number assignment does not need to include a desired port number; in that case, IANA will select from those currently available.

Users should consider whether the requested port number is important. For example, would an assignment be acceptable if IANA picked the port number value? Would a TCP (or other transport protocol) port number assignment be useful by itself? If so, a port number can be assigned to a service for one transport protocol where it is already (or can be subsequently) assigned to a different service for other transport protocols.

The most critical issue in picking a number is selecting the desired range, i.e., System vs. User port numbers. The distinction was intended to indicate a difference in privilege; originally, System port numbers required privileged ('root') access, while User port numbers did not. That distinction has since blurred because some current systems do not limit access control to System port numbers and because some System services have been replicated on User numbers (e.g., IRC). Even so, System port number assignments have continued at an average rate of 3-4 per year over the past 7 years (2007-2013), indicating that the desire to keep this distinction continues.

As a result, the difference between System and User port numbers needs to be treated with caution. Developers are advised to treat services as if they are always run without privilege.

Even when developers seek a System port number assignment, it may be very difficult to obtain. System port number assignment requires IETF Review or IESG Approval and justification that both User and

Dynamic port number ranges are insufficient [RFC6335]. Thus this document recommends both:

>> Developers SHOULD NOT apply for System port number assignments because the increased privilege they are intended to provide is not always enforced.

>> System implementers SHOULD enforce the need for privilege for processes to listen on System port numbers.

At some future date, it might be useful to deprecate the distinction between System and User port numbers altogether. Services typically require elevated ('root') privileges to bind to a System port number, but many such services go to great lengths to immediately drop those privileges just after connection or other association establishment to reduce the impact of an attack using their capabilities. Such services might be more securely operated on User port numbers than on System port numbers. Further, if System port numbers were no longer assigned, as of 2014 it would cost only 180 of the 1024 System values (17%), or 180 of the overall 49152 assigned (System and User) values (<0.04%).

7.4. Support for Security

Just as a service is a way to obtain information or processing from a host over a network, a service can also be the opening through which to compromise that host. Protecting a service involves security, which includes integrity protection, source authentication, privacy, or any combination of these capabilities. Security can be provided in a number of ways, and thus:

>> New services SHOULD support security capabilities, either directly or via a content protection such as TLS [RFC5246] or DTLS [RFC6347] or transport protection such as TCP-AO [RFC5925]. Insecure versions of new or existing secure services SHOULD be avoided because of the new vulnerability they create.

Secure versions of legacy services that are not already security-capable via in-band negotiations can be very useful. However, there is no IETF consensus on when separate ports should be used for secure and insecure variants of the same service [RFC2595] [RFC2817] [RFC6335]. The overall preference is for use of a single port, as noted in Section 6 of this document and Section 7.2 of [RFC6335], but the appropriate approach depends on the specific characteristics of the service. As a result:

>> When requesting both secure and insecure port assignments for the same service, justification is expected for the utility and safety of each port as an independent service (Section 6). Precedent (e.g., citing other protocols that use a separate insecure port) is inadequate justification by itself.

It's also important to recognize that port number assignment is not itself a guarantee that traffic using that number provides the corresponding service, or that a given service is always offered only on its assigned port number. Port numbers are ultimately meaningful only between endpoints and any service can be run on any port. Thus:

>> Security SHOULD NOT rely on assigned port number distinctions alone; every service, whether secure or not, is likely to be attacked.

Applications for a new service that requires both a secure and insecure port may be found, on expert review, to be unacceptable, and may not be approved for allocation. Similarly, an application for a new port to support an insecure variant of an existing secure protocol may be found unacceptable. In both cases, the resulting security of the service in practice will be a significant consideration in the decision as to whether to assign an insecure port.

7.5. Support for Future Versions

Requests for assigned port numbers are expected to support multiple versions on the same assigned port number [RFC6335]. Versions are typically indicated in-band, either at the beginning of a connection or other association, or in each protocol message.

>> Version support SHOULD be included in new services rather than relying on different port number assignments for different versions.

>> Version numbers SHOULD NOT be included in either the service name or service description, to avoid the need to make additional port number assignments for future variants of a service.

Again, the assigned port number space is far too limited to be used as an indicator of protocol version or message type. Although this has happened in the past (e.g., for NFS), it should be avoided in new requests.

7.6. Transport Protocols

IANA assigns port numbers specific to one or more transport protocols, typically UDP [RFC768] and TCP [RFC793], but also SCTP [RFC4960], DCCP [RFC4340], and any other standard transport protocol. Originally, IANA port number assignments were concurrent for both UDP and TCP, and other transports were not indicated. However, to conserve the assigned port number space and to reflect increasing use of other transports, assignments are now specific only to the transport being used.

In general, a service should request assignments for multiple transports using the same service name and description on the same port number only when they all reflect essentially the same service. Good examples of such use are DNS and NFS, where the difference between the UDP and TCP services are specific to supporting each transport. E.g., the UDP variant of a service might add sequence numbers and the TCP variant of the same service might add in-band message delimiters. This document does not describe the appropriate selection of a transport protocol for a service.

>> Service names and descriptions for multiple transport port number assignments SHOULD match only when they describe the same service, excepting only enhancements for each supported transport.

When the services differ, it may be acceptable or preferable to use the same port number, but the service names and descriptions should be different for each transport/service pair, reflecting the differences in the services. E.g., if TCP is used for the basic control protocol and UDP for an alarm protocol, then the services might be "name-ctl" and "name-alarm". A common example is when TCP is used for a service and UDP is used to determine whether that service is active (e.g., via a unicast, broadcast, or multicast test message) [RFC1122]. IANA has, for several years, used the suffix "-disc" in service names to distinguish discovery services, such as are used to identify endpoints capable of a given service:

>> Names of discovery services SHOULD use an identifiable suffix; the suggestion is "-disc".

Some services are used for discovery, either in conjunction with a TCP service or as a stand-alone capability. Such services will be more reliable when using multicast rather than broadcast (over IPv4) because IP routers do not forward "all nodes" broadcasts (all 1's, i.e., 255.255.255.255 for IPv4) and have not been required to support subnet-directed broadcasts since 1999 [RFC1812] [RFC2644].

This issue is relevant only for IPv4 because IPv6 does not support broadcast.

>> UDP over IPv4 multi-host services SHOULD use multicast rather than broadcast.

Designers should be very careful in creating services over transports that do not support congestion control or error recovery, notably UDP. There are several issues that should be considered in such cases, as summarized in Table 1 in [RFC5405]. In addition, the following recommendations apply to service design:

>> Services that use multipoint communication SHOULD be scalable, and SHOULD NOT rely solely on the efficiency of multicast transmission for scalability.

>> Services SHOULD NOT use UDP as a performance enhancement over TCP, e.g., to circumnavigate TCP's congestion control.

7.7. When to Request an Assignment

Assignments are typically requested when a user has enough information to reasonably answer the questions in the IANA application. IANA applications typically take up to a few weeks to process, with some complex cases taking up to a month. The process typically involves a few exchanges between the IANA Ports Expert Review team and the applicant.

An application needs to include a description of the service, as well as to address key questions designed to help IANA determine whether the assignment is justified. The application should be complete and not refer solely to the Internet Draft, RFC, a website, or any other external documentation.

Services that are independently developed can be requested at any time, but are typically best requested in the last stages of design and initial experimentation, before any deployment has occurred that cannot easily be updated.

>> Users MUST NOT deploy implementations that use assigned port numbers prior their assignment by IANA.

>> Users MUST NOT deploy implementations that default to using the experimental System port numbers (1021 and 1022 [RFC4727]) outside a controlled environment where they can be updated with a subsequent assigned port [RFC3692].

Deployments that use unassigned port numbers before assignment complicate IANA management of the port number space. Keep in mind that this recommendation protects existing assignees, users of current services, and applicants for new assignments; it helps ensure that a desired number and service name are available when assigned. The list of currently unassigned numbers is just that - **currently** unassigned. It does not reflect pending applications. Waiting for an official IANA assignment reduces the chance that an assignment request will conflict with another deployed service.

Applications made through Internet Draft / RFC publication (in any stream) typically use a placeholder ("PORTNUM") in the text, and implementations use an experimental port number until a final assignment has been made [RFC6335]. That assignment is initially indicated in the IANA Considerations section of the document, which is tracked by the RFC Editor. When a document has been approved for publication, that request is forwarded to IANA for handling. IANA will make the new assignment accordingly. At that time, IANA may also request that the applicant fill out the application form on their website, e.g., when the RFC does not directly address the information expected as per [RFC6335]. "Early" assignments can be made when justified, e.g., for early interoperability testing, according to existing process [RFC7120] [RFC6335].

>> Users writing specifications SHOULD use symbolic names for port numbers and service names until an IANA assignment has been completed. Implementations SHOULD use experimental port numbers during this time, but those numbers MUST NOT be cited in documentation except as interim.

7.8. Squatting

"Squatting" describes the use of a number from the assignable range in deployed software without IANA assignment for that use, regardless of whether the number has been assigned or remains available for assignment. It is hazardous because IANA cannot track such usage and thus cannot avoid making legitimate assignments that conflict with such unauthorized usage.

Such "squatted" port numbers remain unassigned, and IANA retains the right to assign them when requested by other applicants. Application and service designers are reminded that it is never appropriate to use port numbers that have not been directly assigned [RFC6335]. In particular, any unassigned code from the assigned ranges will be assigned by IANA, and any conflict will be easily resolved as the protocol designer's fault once that happens (because they would not be the assignee). This may reflect in the public's judgment on the

quality of their expertise and cooperation with the Internet community.

Regardless, there are numerous services that have squatted on such numbers that are in widespread use. Designers who are using such port numbers are encouraged to apply for an assignment. Note that even widespread de-facto use may not justify a later IANA assignment of that value, especially if either the value has already been assigned to a legitimate applicant or if the service would not qualify for an assignment of its own accord.

7.9. Other Considerations

As noted earlier, System port numbers should be used sparingly, and it is better to avoid them altogether. This avoids the potentially incorrect assumption that the service on such port numbers run in a privileged mode.

Assigned port numbers are not intended to be changed; this includes the corresponding service name. Once deployed, it can be very difficult to recall every implementation, so the assignment should be retained. However, in cases where the current assignee of a name or number has reasonable knowledge of the impact on such uses, and is willing to accept that impact, the name or number of an assignment can be changed [RFC6335]

Aliases, or multiple service names for the same assigned port number, are no longer considered appropriate [RFC6335].

8. Security Considerations

This document focuses on the issues arising when designing services that require new port assignments. Section 7.4 addresses the security and security-related issues of that interaction.

When designing a secure service, the use of TLS [RFC5246], DTLS [RFC6347], or TCP-AO [RFC5925] mechanisms that protect transport protocols or their contents is encouraged. It may not be possible to use IPsec [RFC4301] in similar ways because of the different relationship between IPsec and port numbers and because applications may not be aware of IPsec protections.

This document reminds application and service designers that port numbers do not protect against denial of service attack or guarantee that traffic should be trusted. Using assigned numbers for port filtering isn't a substitute for authentication, encryption, and integrity protection. The port number alone should not be used to

avoid denial of service attacks or to manage firewall traffic because the use of port numbers is not regulated or validated.

The use of assigned port numbers is the antithesis of privacy because they are intended to explicitly indicate the desired application or service. Strictly, port numbers are meaningful only at the endpoints, so any interpretation elsewhere in the network can be arbitrarily incorrect. However, those numbers can also expose information about available services on a given host. This information can be used by intermediate devices to monitor and intercept traffic as well as to potentially identify key endpoint software properties ("fingerprinting"), which can be used to direct other attacks.

9. IANA Considerations

The entirety of this document focuses on suggestions that help ensure the conservation of port numbers and provide useful hints for issuing informative requests thereof.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2780] Bradner, S., and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.
- [RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3962, Jan. 2004.
- [RFC4727] Fenner, B., "Experimental Values in IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, November 2006.
- [RFC5246] Dierks, T., and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5405] Eggert, L., and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, Nov. 2008.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

- [RFC6335] Cotton, M., L. Eggert, J. Touch, M. Westerlund, and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.
- [RFC6347] Rescorla, E., and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

10.2. Informative References

- [IEN112] Postel, J., "Transmission Control Protocol", IEN 112, August 1979.
- [RFC33] Crocker, S., "New Host-Host Protocol", RFC 33 February 1970.
- [RFC37] Crocker, S., "Network Meeting Epilogue", RFC 37, March 1970.
- [RFC38] Wolfe, S., "Comments on Network Protocol from NWG/RFC #36", RFC 38, March 1970.
- [RFC48] Postel, J., and S. Crocker, "Possible protocol plateau", RFC 48, April 1970.
- [RFC61] Walden, D., "Note on Interprocess Communication in a Resource Sharing Computer Network", RFC 61, July 1970.
- [RFC76] Bouknight, J., J. Madden, and G. Grossman, "Connection by name: User oriented protocol", RFC 76, October 1970.
- [RFC333] Bressler, R., D. Murphy, and D. Walden. "Proposed experiment with a Message Switching Protocol", RFC 333, May 1972.
- [RFC739] Postel, J., "Assigned numbers", RFC 739, November 1977.
- [RFC758] Postel, J., "Assigned numbers", RFC 758, August 1979.
- [RFC768] Postel, J., "User Datagram Protocol", RFC 768, August 1980.
- [RFC793] Postel, J., "Transmission Control Protocol" RFC 793, September 1981
- [RFC820] Postel, J., "Assigned numbers", RFC 820, August 1982.

- [RFC900] Reynolds, J., and J. Postel, "Assigned numbers", RFC 900, June 1984.
- [RFC959] Postel, J., and J. Reynolds, "FILE TRANSFER PROTOCOL (FTP)", RFC 959, October 1985.
- [RFC1122] Braden, B. (Ed.), "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989.
- [RFC1340] Reynolds, J., and J. Postel, "Assigned numbers", RFC 1340, July 1992.
- [RFC1700] Reynolds, J., and J. Postel, "Assigned numbers", RFC 1700, October 1994.
- [RFC1812] Baker, F. (Ed.), "Requirements for IP Version 4 Routers", RFC 1812, June 1995.
- [RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2", RFC 1833, August 1995.
- [RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999.
- [RFC2644] Senie, D., "Changing the Default for Directed Broadcasts in Routers", RFC 2644, August 1999.
- [RFC2817] Khare, R., and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.
- [RFC3232] Reynolds, J. (Ed.), "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, January 2002.
- [RFC3261] Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC4301] Kent, S., and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4340] Kohler, E., M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC4960] Stewart, R. (Ed.), "Stream Control Transmission Protocol", RFC 4960, September 2007.

- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5389] Rosenberg, J., R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT", RFC 5389, October 2008.
- [RFC5766] Mahy, R., P. Matthews, and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.
- [RFC6762] Cheshire, S., and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.
- [RFC6763] Cheshire, S., and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.
- [RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, January 2014.
- [RFC7230] Fielding, R., (Ed.), and J. Reshke, (Ed.), "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

11. Acknowledgments

This work benefitted from the feedback from David Black, Lars Eggert, Gorry Fairhurst, and Eliot Lear, as well as discussions of the IETF TSVWG WG.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292-6695
U.S.A.

Phone: +1 (310) 448-9151
EMail: touch@isi.edu

Network WG
Internet-Draft
Expires: January 4, 2015
Intended Status: Standards Track
Updates: RFC 2872 (if accepted)

James Polk
Subha Dhesikan
Cisco Systems
July 4, 2014

Resource Reservation Protocol (RSVP) Application-ID
Profiles for Voice and Video Streams
draft-ietf-tsvwg-rsvp-app-id-vv-profiles-02

Abstract

RFC 2872 defines an Resource Reservation Protocol (RSVP) object for application identifiers. This document uses that App-ID and gives implementers specific guidelines for differing voice and video stream identifications to nodes along a reservation path, creating specific profiles for voice and video session identification.

Status of this Memo

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

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

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

This Internet-Draft will expire on July 4, 2014.

Copyright Notice

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

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

Table of Contents

1.	Introduction	2
2.	RSVP Application-ID Template	3
3.	The Voice and Video Application-ID Profiles	4
3.1	The Broadcast video Profile	4
3.2	The Real-time Interactive Profile	5
3.3	The Multimedia Conferencing Profile	5
3.4	The Multimedia Streaming Profile	6
3.5	The Conversational Profile	6
4.	Security considerations	7
5.	IANA considerations	7
5.1	Application Profiles	7
5.1.1	Broadcast Profiles IANA Registry	8
5.1.2	Realtime-Interactive Profiles IANA Registry	8
5.1.3	Multimedia-Conferencing Profiles IANA Registry	9
5.1.4	Multimedia-Streaming Profiles IANA Registry	10
5.1.5	Conversational Profiles IANA Registry	10
6.	Acknowledgments	12
7.	References	12
7.1.	Normative References	12
7.2.	Informative References	13
	Authors' Addresses	13
	Appendix	14

1. Introduction

RFC 2872 [RFC2872] describes the usage of policy elements for providing application information in Resource Reservation Protocol (RSVP) signaling [RFC2205]. The intention of providing this information is to enable application-based policy control. However, RFC 2872 does not enumerate any application profiles. The absence of explicit, uniform profiles leads to incompatible handling of these values and misapplied policies. An application profile used by a sender might not be understood by the intermediaries or receiver in a different domain. Therefore, there is a need to enumerate application profiles that are universally understood and applied for correct policy control.

Call control between endpoints has the ability to bind or associate many attributes to a reservation. One new attribute is currently being defined so as to establish the type of traffic contained in that reservation. This is accomplished via assigning a traffic label to the call (or session or flow) [ID-TRAF-CLASS].

This document takes the application traffic classes from [ID-TRAF-CLASS] and places those strings in the APP-ID object defined in RFC 2872. Thus, the intermediary devices (e.g., routers) processing the RSVP message can learn the identified profile within the Application-ID policy element for a particular reservation, and possibly be configured with the profile(s) to understand them

correctly, thus performing the correct admission control.

Another goal of this document is to the ability to signal an application profile which can then be translated into a DSCP value as per the choice of each domain. While the DCLASS object [RFC2996] allows the transfer of DSCP value in an RSVP message, that RFC does not allow the flexibility of having different domains choosing the DSCP value for the traffic classes that they maintain.

How these labels indicate the appropriate Differentiated Services Codepoint (DSCP) is out of scope for this document.

This document will break out each application type and propose how the values in application-id template should be populated for uniformity and interoperability.

1.1 Terminology

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

2. RSVP Application ID Template

The template from RFC 2872 is as follows:

0	1	2	3
PE Length (8)	P-type = AUTH_APP		
Attribute Length	A-type =	Sub-type =	
	POLICY_LOCATOR	ASCII_DN	
Application name as ASCII string (e.g. SAP.EXE)			

In line with how this policy element is constructed in RFC 2872, the A-type will remain "POLICY_LOCATOR".

The P-type field is first created in [RFC2752]. This document uses the existing P-type "AUTH_APP" for application traffic class.

The first Sub-type will be mandatory for every profile within this document, and will be "ASCII_DN". No other Sub-types are defined by any profile within this document, but MAY be included by individual implementations - and MUST be ignored if not understood by receiving implementations along the reservation path.

RFC 2872 states the #1 sub-element from RFC 2872 as the "identifier that uniquely identifies the application vendor", which is optional to include. This document modifies this vendor limitation so that the identifier need only be unique - and not limited to an application vendor (identifier). For example, this specification now allows an RFC that defines an industry recognizable term or string to be a valid identifier. For example, a term or string taken from another IETF document, such as "conversational" or "avconf" from [ID-TRAF-CLASS]. This sub-element is still optional to include.

The following subsections will define the values within the above template into specific profiles for voice and video identification.

3. The Voice and Video Application-ID Profiles

This section contains the elements of the Application ID policy object which is used to signal the application classes defined in [ID-TRAF-CLASS].

3.1 The Broadcast Profiles

Broadcast profiles are for minimally buffered one-way streaming flows, such as video surveillance, or Internet based concerts or non-VOD TV broadcasts such as live sporting events.

This document creates Broadcast profiles for

- Broadcast IPTV for audio and video
- Broadcast Live-events for audio and video
- Broadcast Surveillance for audio and video

Here is an example profile for identifying Broadcast Video-Surveillance

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,  
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
APP=broadcast.video.surveillance, VER="
```

[Editor's Note: "rfcXXXX" will be replaced with the RFC number assigned to the [ID-TRAF-CLASS] reference. This 'note' should be removed during the RFC-Editor review process.]

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value at this time.

3.2 The Realtime Interactive Profiles

Realtime Interactive profiles are for on-line gaming, and both remote and virtual avconf applications, in which the timing is particularly important towards the feedback to uses of these applications. This traffic type will generally not be UDP based, with minimal tolerance to RTT delays.

This document creates Realtime Interactive profiles for

- Realtime-Interactive Gaming
- Realtime-Interactive Remote-Desktop
- Realtime-Interactive Virtualized-Desktop

Here is the profile for identifying Realtime-Interactive Gaming

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,  
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
APP=realtime-interactive.gaming, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.3 The Multimedia Conferencing Profiles

There will be Multimedia Conferencing profiles for presentation data, application sharing and whiteboarding, where these applications will most often be associated with a larger Conversational (audio and/or audio/video) conference. Timing is important, but some minimal delays are acceptable, unlike the case for Realtime-Interactive traffic.

This document creates Multimedia-Conferencing profiles for

- Multimedia-Conferencing presentation-data
- Multimedia-Conferencing presentation-video
- Multimedia-Conferencing presentation-audio
- Multimedia-Conferencing application-sharing
- Multimedia-Conferencing whiteboarding

Here is the profile for identifying Multimedia-Conferencing Application-sharing

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,  
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
APP=multimedia-conferencing.application-sharing, VER="
```

Where the Globally Unique Identifier (GUID) indicates the RFC reference that created this well-known string [ID-TRAF-CLASS], the

APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.4 The Multimedia Streaming Profiles

Multimedia Streaming profiles are for more significantly buffered one-way streaming flows than Broadcast profiles. These include...

This document creates Multimedia Streaming profiles for

- Multimedia-Streaming multiplex
- Multimedia-Streaming webcast

Here is the profile for identifying Multimedia Streaming webcast

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,  
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
APP=multimedia-streaming.webcast, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

3.5 The Conversational Profiles

Conversational category is for realtime bidirectional communications, such as voice or video, and is the most numerous due to the choices of application with or without adjectives. The number of profiles is then doubled because there needs to be one for unadmitted and one for admitted. The IANA section lists all that are currently proposed for registration at this time, therefore there will not be an exhaustive list provided in this section.

This document creates Conversational profiles for

- Conversational Audio
- Conversational Audio Admitted
- Conversational Video
- Conversational Video Admitted
- Conversational Audio Avconf
- Conversational Audio Avconf Admitted
- Conversational Video Avconf
- Conversational Video Avconf Admitted
- Conversational Audio Immersive
- Conversational Audio Immersive Admitted
- Conversational Video Immersive
- Conversational Video Immersive Admitted

Here is an example profile for identifying Conversational Audio:

```
AUTH_APP, POLICY_LOCATOR, ASCII_DN,  
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
APP=conversational.audio, VER="
```

Where the Globally Unique Identifier (GUID) indicates the documented reference that created this well-known string [ID-TRAF-CLASS], the APP is the profile name with no spaces, and the "VER=" is included, but has no value, but MAY if versioning becomes important.

4. Security considerations

The security considerations section within RFC 2872 sufficiently covers this document, with one possible exception - someone using the wrong template values (e.g., claiming a reservation is Multimedia Streaming when it is in fact Real-time Interactive). Given that each traffic flow is within separate reservations, and RSVP does not have the ability to police the type of traffic within any reservation, solving for this appears to be administratively handled at best. This is not meant to be a 'punt', but there really is nothing this template creates that is going to make things any harder for anyone (that we know of now).

5. IANA considerations

5.1 Application Profiles

This document requests IANA create a new registry for the application identification classes similar to the following table within the Resource Reservation Protocol (RSVP) Parameters registry:

```
Registry Name: RSVP APP-ID Profiles  
Reference: [this document]  
Registration procedures: Standards Track document [RFC5226]
```

```
[Editor's Note: "rfcXXXX" will be replaced with the RFC number  
assigned to the [ID-TRAF-CLASS] reference. This  
'note' should be removed during the RFC-Editor  
review process.]
```

5.1.1 Broadcast Profiles IANA Registry

Broadcast Audio IPTV Profile

```
P-type = AUTH_APP  
A-type = POLICY_LOCATOR  
Sub-type = ASCII_DN  
Conformant policy locator =  
    "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,  
    APP=broadcast.audio.iptv, VER="
```

Reference: [this document]

Broadcast Video IPTV Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.video.iptv, VER="

Reference: [this document]

Broadcast Audio Live-events Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.audio.live-events, VER="

Reference: [this document]

Broadcast Video Live-events Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.video.live-events, VER="

Reference: [this document]

Broadcast Audio-Surveillance Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.audio.surveillance, VER="

Reference: [this document]

Broadcast Video-Surveillance Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=broadcast.video.surveillance, VER="

Reference: [this document]

5.1.2 Realtime-Interactive Profiles IANA Registry

Realtime-Interactive Gaming Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP= realtime-interactive.gaming, VER="

Reference: [this document]

Real-time Interactive Remote-Desktop Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=realtime-interactive.remote-desktop, VER="

Reference: [this document]

Real-time Interactive Virtualized-Desktop Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=realtime-interactive.
remote-desktop.virtual, VER="

Reference: [this document]

Real-time Interactive Telemetry Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=realtime-interactive.telemetry, VER="

Reference: [this document]

5.1.3 Multimedia-Conferencing Profiles IANA Registry

Multimedia-Conferencing Presentation-Data Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP= multimedia-conferencing.presentation-data,
VER="

Reference: [this document]

Multimedia-Conferencing Presentation-Video Profile

P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,

APP= multimedia-conferencing.presentation-video,
VER="

Reference: [this document]

Multimedia-Conferencing Presentation-Audio Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP= multimedia-conferencing.presentation-audio,
VER="

Reference: [this document]

Multimedia-Conferencing Application-Sharing Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP= multimedia-conferencing.application-sharing,
VER="

Reference: [this document]

Multimedia-Conferencing Whiteboarding Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP= multimedia-conferencing.whiteboarding, VER="

Reference: [this document]

5.1.4 Multimedia-Streaming Profiles IANA Registry

Multimedia-Streaming Multiplex Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=multimedia-streaming.multiplex, VER="

Reference: [this document]

Multimedia-Streaming Webcast Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=multimedia-streaming.webcast, VER="

Reference: [this document]

5.1.5 Conversational Profiles IANA Registry

Conversational Audio Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.audio, VER="

Reference: [this document]

Conversational Audio Admitted Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.audio.aq:admitted, VER="

Reference: [this document]

Conversational Video Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.video, VER="

Reference: [this document]

Conversational Video Admitted Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.video.aq:admitted, VER="

Reference: [this document]

Conversational Audio Avconf Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.audio.avconf, VER="

Reference: [this document]

Conversational Audio Avconf Admitted Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
 APP=conversational.audio.avconf.aq:admitted,
 VER="

Reference: [this document]

Conversational Video Avconf Profile
P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
 APP=conversational.video.avconf, VER="

Reference: [this document]

Conversational Video Avconf Admitted Profile
P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
 APP=conversational.video.avconf.aq:admitted,
 VER="

Reference: [this document]

Conversational Audio Immersive Profile
P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
 APP=conversational.audio.immersive, VER="

Reference: [this document]

Conversational Audio Immersive Admitted Profile
P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
 APP=conversational.audio.immersive.aq:admitted,
 VER="

Reference: [this document]

Conversational Video Immersive Profile
P-type = AUTH_APP
A-type = POLICY_LOCATOR
Sub-type = ASCII_DN
Conformant policy locator =
 "GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,

APP=conversational.video.immersive, VER="

Reference: [this document]

Conversational Video Immersive Admitted Profile

P-type = AUTH_APP

A-type = POLICY_LOCATOR

Sub-type = ASCII_DN

Conformant policy locator =

"GUID=http://www.rfc-editor.org/rfc/rfcXXXX.txt,
APP=conversational.video.immersive.aq:admitted,
VER="

Reference: [this document]

6. Acknowledgments

To Francois Le Faucheur, Paul Jones, Ken Carlberg, Georgios Karagiannis and Glen Lavers for their helpful comments, document reviews and encouragement.

7. References

7.1. Normative References

- [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997
- [RFC2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997
- [RFC2474] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers ", RFC 2474, December 1998
- [RFC2750] S. Herzog, "RSVP Extensions for Policy Control", RFC 2750, January 2000
- [RFC2872] Y. Bernet, R. Pabbati, "Application and Sub Application Identity Policy Element for Use with RSVP", RFC 2872, June 2000
- [RFC2996] Y. Bernet, "Format of the RSVP DCLASS Object", RFC 2996, November 2000
- [RFC3182] S. Yadav, R. Yavatkar, R. Pabbati, P. Ford, T. Moore, S. Herzog, R. Hess, "Identity Representation for RSVP", RFC 3182, October 2001
- [RFC5226] T. Narten, H. Alvestrand, "Guidelines for Writing an IANA

Considerations Section in RFCs", RFC 5226, May 2008

[ID-TRAF-CLASS] J. Polk, S. Dhesikan, P. Jones, "The Session Description Protocol (SDP) 'trafficclass' Attribute", work in progress, Feb 2013

7.2. Informative References

[RFC4594] J. Babiarz, K. Chan, F Baker, "Configuration Guidelines for Diffserv Service Classes", RFC 4594, August 2006

Authors' Addresses

James Polk
3913 Treemont Circle
Colleyville, Texas, USA
+1.817.271.3552

mailto: jmpolk@cisco.com

Subha Dhesikan
170 W Tasman St
San Jose, CA, USA
+1.408-902-3351

mailto: sdhesika@cisco.com

Appendix - Changes to ID

[Editor's Note: this appendix should be removed in the RFC-Editor's process.]

A.1 - Changes from WG version -00 to WG version -01

The following changes were made in this version:

- corrected nits
- globally replaced GUID link from the MMUSIC Trafficclass ID to the future RFC of that document.
- added profiles for presentation-video and presentation-audio

A.2 - Changes from Individual -04 to WG version -00

The following changes were made in this version:

- changed P-Type from APP_TC back to AUTH_APP, which is already defined.
- fixed nits and inconsistencies

A.3 - Changes from Individual -03 to -04

The following changes were made in this version:

- clarified security considerations section to mean RSVP cannot police the type of traffic within a reservation to know if a traffic flow should be using a different profile, as defined in this document.
- changed existing informative language regarding "... other Sub-types ..." from 'can' to normative 'MAY'.
- editorial changes to clear up minor mistakes

A.4 - Changes from Individual -02 to -03

The following changes were made in this version:

- Added [ID-TRAF-CLASS] as a reference
- Changed to a new format of the profile string.
- Added many new profiles based on the new format into each parent category of Section 3.
- changed the GUID to refer to draft-ietf-mmusic-traffic-class-for-sdp-03.txt
- changed 'desktop' adjective to 'avconf' to keep in alignment with [ID-TRAF-CLASS]
- Have a complete IANA Registry proposal for each application-ID discussed in this draft.
- General text clean-up of the draft.

Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: April 6, 2015

Georgios Karagiannis
Huawei Technologies
Anurag Bhargava
Cisco Systems, Inc.
October 6, 2014

Generic Aggregation of Resource ReSerVation Protocol (RSVP)
for IPv4 And IPv6 Reservations over PCN domains
draft-ietf-tsvwg-rsvp-pcn-11

Abstract

This document specifies extensions to Generic Aggregated RSVP RFC 4860 for support of the PCN Controlled Load (CL) and Single Marking (SM) edge behaviors over a Diffserv cloud using Pre-Congestion Notification.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 6, 2015.

Copyright Notice

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

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

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].

Table of Contents

1. Introduction	4
1.1. Objective	4
1.2. Overview and Motivation	5
1.3. Terminology	7
1.4. Organization of This Document	11
2. Overview of RSVP extensions and Operations	11
2.1. Overview of RSVP Aggregation Procedures in PCN domains	11
2.2. PCN Marking and encoding and transport of pre-congestion Information	13
2.3. Traffic Classification Within The Aggregation Region	13
2.4. Deaggregator (PCN-egress-node) Determination	13
2.5. Mapping E2E Reservations Onto Aggregate Reservations	13
2.6. Size of Aggregate Reservations	14
2.7. E2E Path ADSPEC update	14
2.8. Intra-domain Routes	14
2.9. Inter-domain Routes	15
2.10. Reservations for Multicast Sessions	15
2.11. Multi-level Aggregation	15
2.12. Reliability Issues	15
3. Elements of Procedure	15
3.1. Receipt of E2E Path Message by PCN-ingress-node (aggregating router)	15
3.2. Handling Of E2E Path Message by Interior Routers	16
3.3. Receipt of E2E Path Message by PCN-egress-node (deaggregating router)	16
3.4. Initiation of new Aggregate Path Message By PCN-ingress-node (Aggregating Router)	16
3.5. Handling Of new Aggregate Path Message by Interior Routers	16
3.6. Handling Of Aggregate Path Message by Deaggregating Router	16
3.7. Handling of E2E Resv Message by Deaggregating Router	17
3.8. Handling Of E2E Resv Message by Interior Routers	17

3.9. Initiation of New Aggregate Resv Message By Deaggregating Router	17
3.10. Handling of Aggregate Resv Message by Interior Routers	18
3.11. Handling of E2E Resv Message by Aggregating Router	18
3.12. Handling of Aggregated Resv Message by Aggregating Router . .	18
3.13. Removal of E2E Reservation	19
3.14. Removal of Aggregate Reservation	19
3.15. Handling of Data On Reserved E2E Flow by Aggregating Router .	19
3.16. Procedures for Multicast Sessions	19
3.17. Misconfiguration of PCN node	19
3.18. PCN based Flow Termination	19
4. Protocol Elements	20
4.1 PCN object	20
5. Security Considerations	23
6. IANA Considerations	24
7. Acknowledgments	24
8. Normative References	24
9. Informative References	25
10. Appendix A: Example Signaling Flow	26
11. Authors' Address	29

1. Introduction

1.1 Objective

Pre-Congestion Notification (PCN) can support the quality of service (QoS) of inelastic flows within a Diffserv domain in a simple, scalable, and robust fashion. Two mechanisms are used: admission control and flow termination. Admission control is used to decide whether to admit or block a new flow request, while flow termination is used in abnormal circumstances to decide whether to terminate some of the existing flows. To support these two features, the overall rate of PCN-traffic is metered on every link in the domain, and PCN-packets are appropriately marked when certain configured rates are exceeded. These configured rates are below the rate of the link, thus providing notification to boundary nodes about overloads before any congestion occurs (hence "pre-congestion" notification). The PCN-egress-nodes measure the rates of differently marked PCN traffic in periodic intervals and report these rates to the Decision Points for admission control and flow termination; the Decision Points use these rates to make decisions. The Decision Points may be collocated with the PCN-ingress-nodes, or their function may be implemented in a another node. For more details see [RFC5559], [RFC6661], and [RFC6662].

The main objective of this document is to specify the signaling protocol that can be used within a Pre-Congestion Notification (PCN) domain to carry reports from a PCN-ingress-node to a PCN Decision point, considering that the PCN Decision Point and PCN-egress-node are collocated.

If the PCN Decision Point is not collocated with the PCN-egress-node then additional signaling procedures are required that are out of the scope of this document. Moreover, as mentioned above this architecture conforms with PBAC (Policy-Based Admission Control), when the Decision Point is located in a another node then the PCN-ingress-node [RFC2753].

Several signaling protocols can be used to carry information between PCN-boundary-nodes (PCN-ingress-node and PCN-egress-node). However, since (1) both PCN-egress-node and PCN-ingress-nodes are located on the data path and (2) the admission control procedure needs to be done at PCN-egress-node, a signaling protocol that follows the same path as the data path, like RSVP (Resource Reservation Protocol), is more suited for this purpose. In particular, this document specifies extensions to Generic Aggregated RSVP [RFC4860] for support of the PCN Controlled Load (CL) and Single Marking (SM) edge behaviors over a Diffserv cloud using Pre-Congestion Notification.

This draft is intended to be published as Experimental in order to:

- o) validate industry interest by allowing implementation and deployment
- o) gather operational experience, in particular around dynamic interactions of RSVP signaling and PCN notification and

corresponding levels of performance.

Support for the techniques specified in this document involves RSVP functionality in boundary nodes of a PCN domain whose interior nodes forward RSVP traffic without performing RSVP functionality.

1.2 Overview and Motivation

Two main Quality of Service (QoS) architectures have been specified by the IETF. These are the Integrated Services (Intserv) [RFC1633] architecture and the Differentiated Services (DiffServ) architecture ([RFC2475]).

Intserv provides methods for the delivery of end-to-end Quality of Service (QoS) to applications over heterogeneous networks. One of the QoS signaling protocols used by the Intserv architecture is the Resource reSerVation Protocol (RSVP) [RFC2205], which can be used by applications to request per-flow resources from the network. These RSVP requests can be admitted or rejected by the network. Applications can express their quantifiable resource requirements using Intserv parameters as defined in [RFC2211] and [RFC2212]. The Controlled Load (CL) service [RFC2211] is a quality of service (QoS) closely approximating the QoS that the same flow would receive from a lightly loaded network element. The CL service is useful for inelastic flows such as those used for real-time media.

The DiffServ architecture can support the differentiated treatment of packets in very large scale environments. While Intserv and RSVP classify packets per-flow, Diffserv networks classify packets into one of a small number of aggregated flows or "classes", based on the Diffserv codepoint (DSCP) in the packet IP header. At each Diffserv router, packets are subjected to a "per-hop behavior" (PHB), which is invoked by the DSCP. The primary benefit of Diffserv is its scalability, since the need for per-flow state and per-flow processing, is eliminated.

However, DiffServ does not include any mechanism for communication between applications and the network. Several solutions have been specified to solve this issue. One of these solutions is Intserv over Diffserv [RFC2998] including resource-based admission control (RBAC), PBAC, assistance in traffic identification/classification, and traffic conditioning. Intserv over Diffserv can operate over a statically provisioned or a RSVP aware Diffserv region. When it is RSVP aware, several mechanisms may be used to support dynamic provisioning and topology-aware admission control, including aggregate RSVP reservations, per-flow RSVP, or a bandwidth broker. [RFC3175] specifies aggregation of Resource ReSerVation Protocol (RSVP) end-to-end reservations over aggregate RSVP reservations. In [RFC3175] the RSVP generic aggregated reservation is characterized by a RSVP SESSION object using the 3-tuple <source IP address, destination IP address, Diffserv Code Point>.

Several scenarios require the use of multiple generic aggregate reservations that are established for a given PHB from a given source

IP address to a given destination IP address, see [SIG-NESTED], [RFC4860]. For example, multiple generic aggregate reservations can be applied in the situation that multiple E2E reservations using different preemption priorities need to be aggregated through a PCN-domain using the same PHB. By using multiple aggregate reservations for the same PHB, it allows enforcement of the different preemption priorities within the aggregation region. This allows more efficient management of the Diffserv resources, and in periods of resource shortage, this allows sustainment of a larger number of E2E reservations with higher preemption priorities. In particular, [SIG-NESTED] discusses in detail how end-to-end RSVP reservations can be established in a nested VPN environment through RSVP aggregation.

[RFC4860] provides generic aggregate reservations by extending [RFC3175] to support multiple aggregate reservations for the same source IP address, destination IP address, and PHB (or set of PHBs). In particular, multiple such generic aggregate reservations can be established for a given PHB from a given source IP address to a given destination IP address. This is achieved by adding the concept of a Virtual Destination Port and of an Extended Virtual Destination Port in the RSVP SESSION object. In addition to this, the RSVP SESSION object for generic aggregate reservations uses the PHB Identification Code (PHB-ID) defined in [RFC3140], instead of using the Diffserv Code Point (DSCP) used in [RFC3175]. The PHB-ID is used to identify the PHB, or set of PHBs, from which the Diffserv resources are to be reserved.

The RSVP like signaling protocol required to carry (1) requests from a PCN-egress-node to a PCN-ingress-node and (2) reports from a PCN-ingress-node to a PCN-egress-node needs to follow the PCN signaling requirements defined in [RFC6663]. In addition to that the signaling protocol functionality supported by the PCN-ingress-nodes and PCN-egress-nodes needs to maintain logical aggregate constructs (i.e. ingress-egress-aggregate state) and be able to map E2E reservations to these aggregate constructs. Moreover, no actual reservation state is needed to be maintained inside the PCN domain, i.e., the PCN-interior-nodes are not maintaining any reservation state.

This can be accomplished by two possible approaches:

Approach (1):

- o) adapting the RFC 4860 aggregation procedures to fit the PCN requirements with as little change as possible over the RFC 4860 functionality
- o) hence performing aggregate RSVP signaling (even if it is to be ignored by PCN interior nodes)
- o) using this aggregate RSVP signaling procedures to carry PCN information between the PCN-boundary-nodes (PCN-ingress-node and PCN-egress-node).

Approach (2):

- o) adapting the RFC 4860 aggregation procedures to fit the PCN requirements with more significant changes over RFC4860 (i.e. the aspect of the procedures that have to do with maintaining aggregate states and to do with mapping the E2E reservations to aggregate constructs are kept, but the procedures that have to do with the aggregate RSVP signaling and aggregate reservation establishment/maintenance are dropped).
- o) hence not performing aggregate RSVP signaling
- o) piggy-backing of the PCN information inside the E2E RSVP signaling.

Both approaches are probably viable, however, since the RFC 4860 operations have been thoroughly studied and implemented, it can be considered that the RFC 4860 solution can better deal with the more challenging situations (rerouting in the PCN domain, failure of an PCN-ingress-node, failure of an PCN-egress-node, rerouting towards a different edge, etc.). This is the reason for choosing Approach (1) for the specification of the signaling protocol used to carry PCN information between the PCN-boundary-nodes (PCN-ingress-node and PCN-egress-node).

In particular, this document specifies extensions to Generic Aggregated RSVP [RFC4860] for support of the PCN Controlled Load (CL) and Single Marking (SM) edge behaviors over a Diffserv cloud using Pre-Congestion Notification.

This document follows the PCN signaling requirements defined in [RFC6663] and specifies extensions to Generic Aggregated RSVP [RFC4860] for support of PCN edge behaviors as specified in [RFC6661] and [RFC6662]. Moreover, this document specifies how RSVP aggregation can be used to setup and maintain: (1) Ingress Egress Aggregate (IEA) states at Ingress and Egress nodes and (2) generic aggregation of RSVP end-to-end RSVP reservations over PCN (Congestion and Pre-Congestion Notification) domains.

To comply with this specification, PCN-nodes MUST be able to support the functionality specified in [RFC5670], [RFC5559], [RFC6660], [RFC6661], [RFC6662]. Furthermore, the PCN-boundary-nodes MUST support the RSVP generic aggregated reservation procedures specified in [RFC4860] which are augmented with procedures specified in this document.

1.3. Terminology

This document uses terms defined in [RFC4860], [RFC3175], [RFC5559], [RFC5670], [RFC6661], [RFC6662].

For readability, a number of definitions from [RFC3175] as well as definitions for terms used in [RFC5559], [RFC6661], and [RFC6662] are provided here, where some of them are augmented with new meanings:

- Aggregator** This is the process in (or associated with) the router at the ingress edge of the aggregation region (with respect to the end-to-end RSVP reservation) and behaving in accordance with [RFC4860]. In this document, it is also the PCN-ingress-node. It is important to notice that in the context of this document the Aggregator must be able to determine the Deaggregator using the procedures specified in Section 4 of [RFC4860] and in Section 1.4.2 of [RFC3175].
- Congestion level estimate (CLE):**
The ratio of PCN-marked to total PCN-traffic (measured in octets) received for a given ingress-egress-aggregate during a given measurement period. The CLE is used to derive the PCN-admission-state and is also used by the report suppression procedure if report suppression is activated.
- Deaggregator** This is the process in (or associated with) the router at the egress edge of the aggregation region (with respect to the end-to-end RSVP reservation) and behaving in accordance with [RFC4860]. In this document, it is also the PCN-egress-node and Decision Point.
- E2E** end to end
- E2E Reservation** This is an RSVP reservation such that:
- (i) corresponding RSVP Path messages are initiated upstream of the Aggregator and terminated downstream of the Deaggregator, and
 - (ii) corresponding RSVP Resv messages are initiated downstream of the Deaggregator and terminated upstream of the Aggregator, and
 - (iii) this RSVP reservation is aggregated over an Ingress Egress Aggregate (IEA) between the Aggregator and Deaggregator.
- An E2E RSVP reservation may be a per-flow reservation, which in this document is only maintained at the PCN-ingress-node and PCN-egress-node. Alternatively, the E2E reservation may itself be an aggregate reservation of various types (e.g., Aggregate IP reservation, Aggregate IPsec reservation, see [RFC4860]). As per regular RSVP operations, E2E RSVP reservations are unidirectional.
- E2E microflow** a microflow where its associated packets are being forwarded on an E2E path.

Extended vDstPort (Extended Virtual Destination Port)

An identifier used in the SESSION that remains constant over the life of the generic aggregate reservation. The length of this identifier is 32-bits when IPv4 addresses are used and 128 bits when IPv6 addresses are used.

A sender(or Aggregator) that wishes to narrow the scope of a SESSION to the sender-receiver pair (or Aggregator-Deaggregator pair) should place its IPv4 or IPv6 address here as a network unique identifier. A sender (or Aggregator) that wishes to use a common session with other senders (or Aggregators) in order to use a shared reservation across senders (or Aggregators) must set this field to all zeros. In this document, the Extended vDstPort should contain the IPv4 or IPv6 address of the Aggregator.

ETM-rate

The rate of excess-traffic-marked PCN-traffic received at a PCN-egress-node for a given ingress-egress-aggregate in octets per second.

Ingress-egress-aggregate (IEA):

The collection of PCN-packets from all PCN-flows that travel in one direction between a specific pair of PCN-boundary-nodes. In this document one RSVP generic aggregated reservation is mapped to only one ingress-egress-aggregate, while one ingress-egress-aggregate is mapped to either one or to more than one RSVP generic aggregated reservations. PCN-flows and their PCN-traffic that are mapped into a specific RSVP generic aggregated reservation can also easily be mapped into their corresponding ingress-egress-aggregate.

Microflow:
(from [RFC2474])

a single instance of an application-to-application flow of packets which is identified by source address, destination address, protocol id, and source port, destination port (where applicable).

PCN-domain:

a PCN-capable domain; a contiguous set of PCN-enabled nodes that perform Diffserv scheduling [RFC2474]; the complete set of PCN-nodes that in principle can, through PCN-marking packets, influence decisions about flow admission and termination within the domain; includes the PCN-egress-nodes, which measure these PCN-marks, and the PCN-ingress-nodes.

PCN-boundary-node: a PCN-node that connects one PCN-domain to a node either in another PCN-domain or in a non-PCN-domain.

- PCN-interior-node: a node in a PCN-domain that is not a PCN-boundary-node.
- PCN-node: a PCN-boundary-node or a PCN-interior-node.
- PCN-egress-node: a PCN-boundary-node in its role in handling traffic as it leaves a PCN-domain. In this document the PCN-egress-node operates also as a Decision Point and Deaggregator.
- PCN-ingress-node: a PCN-boundary-node in its role in handling traffic as it enters a PCN-domain. In this document the PCN-ingress-node operates also as a Aggregator.
- PCN-traffic,
PCN-packets,
PCN-BA: a PCN-domain carries traffic of different Diffserv behavior aggregates (BAs) [RFC2474]. The PCN-BA uses the PCN mechanisms to carry PCN-traffic, and the corresponding packets are PCN-packets. The same network will carry traffic of other Diffserv BAs. The PCN-BA is distinguished by a combination of the Diffserv codepoint (DSCP) and ECN fields.
- PCN-flow: the unit of PCN-traffic that the PCN-boundary-node admits (or terminates); the unit could be a single E2E microflow (as defined in [RFC2474]) or some identifiable collection of microflows.
- PCN-admission-state: The state ("admit" or "block") derived by the Decision Point for a given ingress-egress-aggregate based on statistics about PCN-packet marking. The Decision Point decides to admit or block new flows offered to the aggregate based on the current value of the PCN-admission-state.
- PCN-sent-rate The rate of PCN-traffic received at a PCN-ingress-node and destined for a given ingress-egress-aggregate in octets per second.
- PHB-ID (Per Hop Behavior Identification Code)
A 16-bit field containing the Per Hop Behavior Identification Code of the PHB, or of the set of PHBs, from which Diffserv resources are to be reserved. This field must be encoded as specified in Section 2 of [RFC3140].
- RSVP generic aggregated reservation: an RSVP reservation that is identified by using the RSVP SESSION object for generic RSVP aggregated reservation. This RSVP

SESSION object is based on the RSVP SESSION object specified in [RFC4860] augmented with the following information:

- o) the IPv4 DestAddress, IPv6 DestAddress should be set to the IPv4 or IPv6 destination addresses, respectively, of the Deaggregator (PCN-egress-node)
- o) PHB-ID (Per Hop Behavior Identification Code) should be set equal to PCN-compatible Diffserv codepoint(s).
- o) Extended vDstPort should be set to the IPv4 or IPv6 destination addresses, of the Aggregator (PCN-ingress-node)

VDstPort (Virtual Destination Port)

A 16-bit identifier used in the SESSION that remains constant over the life of the generic aggregate reservation.

1.4. Organization of This Document

This document is organized as follows. Section 2 gives an overview of RSVP extensions and operations. The elements of the used procedures are specified in Section 3. Section 4 describes the protocol elements. The security considerations are given in section 5 and the IANA considerations are provided in Section 6.

2. Overview of RSVP extensions and Operations

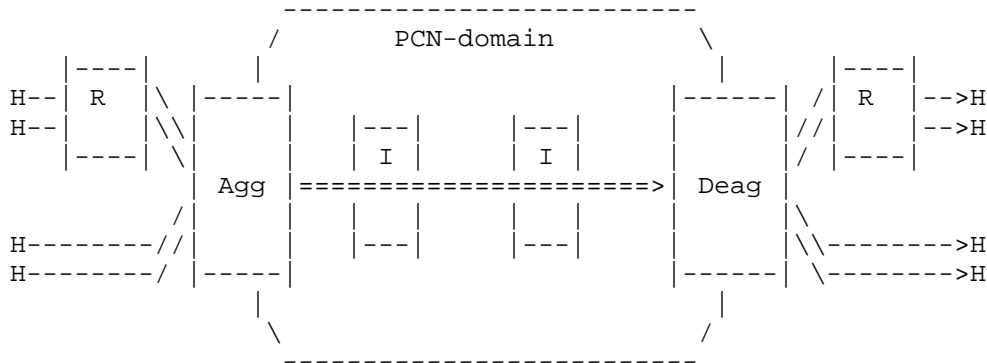
2.1 Overview of RSVP Aggregation Procedures in PCN domains

The PCN-boundary-nodes, see Figure 1, can support RSVP SESSIONS for generic aggregated reservations [RFC4860], which are depending on ingress-egress-aggregates. In particular, one RSVP generic aggregated reservation matches to only one ingress-egress-aggregate.

However, one ingress-egress-aggregate matches to either one, or more than one, RSVP generic aggregated reservations. In addition, to comply with this specification, the PCN-boundary nodes need to distinguish and process (1) RSVP SESSIONS for generic aggregated sessions and their messages according to [RFC4860], (2) E2E RSVP sessions and messages according to [RFC2205].

This document locates all RSVP processing for a PCN domain at PCN-Boundary nodes. PCN-interior-nodes do not perform any RSVP functionality or maintain RSVP-related state information. Rather, PCN-interior nodes forward all RSVP messages (for both generic aggregated reservations[RFC4860] and end to end reservations [RFC2205]) as if they were ordinary network traffic.

Moreover, each Aggregator and Deaggregator (i.e., PCN-boundary-nodes) need to support policies to initiate and maintain for each pair of PCN-boundary-nodes of the same PCN-domain one ingress-egress-aggregate.



H = Host requesting end-to-end RSVP reservations
 R = RSVP router
 Agg = Aggregator (PCN-ingress-node)
 Deag = Deaggregator (PCN-egress-node)
 I = Interior Router (PCN-interior-node)
 --> = E2E RSVP reservation
 ==> = Aggregate RSVP reservation

Figure 1 : Aggregation of E2E Reservations
 over Generic Aggregate RSVP Reservations
 in PCN domains, based on [RFC4860]

Both the Aggregator and Deaggregator can maintain one or more RSVP generic aggregated Reservations, but the Deaggregator is the entity that initiates these RSVP generic aggregated reservations. Note that one RSVP generic aggregated reservation matches to only one ingress-egress-aggregate, while one ingress-egress-aggregate matches to either one or to more than one RSVP generic aggregated reservations. This can be accomplished by using for the different RSVP generic aggregated reservations the same combinations of ingress and egress identifiers, but with a different PHB-ID value (see [RFC4860]). The procedures for aggregation of E2E reservations over generic aggregate RSVP reservations are the same as the procedures specified in Section 4 of [RFC4860], augmented with the ones specified in Section 2.5.

One significant difference between this document and [RFC4860] is the fact that in this document the admission control of E2E RSVP reservations over the PCN core is performed according to the PCN procedures, while in [RFC4860] this is achieved via first admitting aggregate RSVP reservations over the aggregation region and then admitting the E2E reservations over the aggregate RSVP reservations. Therefore, in this document, the RSVP generic aggregate RSVP reservations are not subject to admission control in the PCN-core, and the E2E RSVP reservations are not subject to admission control

over the aggregate reservations. In turn, this means that several procedures of [RFC4860] are significantly simplified in this document:

- o) unlike [RFC4860], the generic aggregate RSVP reservations need not be admitted in the PCN core.
- o) unlike [RFC4860], the RSVP aggregated traffic does not need to be tunneled between Aggregator and Deaggregator, see Section 2.3.
- o) unlike [RFC4860], the Deaggregator need not perform admission control of E2E reservations over the aggregate RSVP reservations.
- o) unlike [RFC4860], there is no need for dynamic adjustment of the RSVP generic aggregated reservation size, see Section 2.6.

2.2 PCN Marking and encoding and transport of pre-congestion information

The method of PCN marking within the PCN domain is specified in [RFC5670]. In addition, the method of encoding and transport of pre-congestion information is specified in [RFC6660]. The PHB-ID (Per Hop Behavior Identification Code) used SHOULD be set equal to PCN-compatible Diffserv codepoint(s).

2.3. Traffic Classification Within The Aggregation Region

The PCN-ingress marks a PCN-BA using PCN-marking (i.e., combination of the DSCP and ECN fields), which interior nodes use to classify PCN-traffic. The PCN-traffic (e.g., E2E microflows) belonging to a RSVP generic aggregated reservation can be classified only at the PCN-boundary-nodes (i.e., Aggregator and Deaggregator) by using the RSVP SESSION object for RSVP generic aggregated reservations, see Section 2.1 of [RFC4860]. Note that the DSCP value included in the SESSION object, SHOULD be set equal to a PCN-compatible Diffserv codepoint. Since no admission control procedures over the RSVP generic aggregated reservations in the PCN-core are required, unlike [RFC4860], the RSVP aggregated traffic need not to be tunneled between Aggregator and Deaggregator. In this document one RSVP generic aggregated reservation is mapped to only one ingress-egress-aggregate, while one ingress-egress-aggregate is mapped to either one or to more than one RSVP generic aggregated reservations. PCN-flows and their PCN-traffic that are mapped into a specific RSVP generic aggregated reservation can also easily be classified into their corresponding ingress-egress-aggregate. The method of traffic conditioning of PCN-traffic and non-PCN traffic and PHB configuration is described in [RFC6661] and [RFC6662].

2.4. Deaggregator Determination

The present document assumes the same dynamic Deaggregator determination method as used in [RFC4860].

2.5. Mapping E2E Reservations Onto Aggregate Reservations

To comply with this specification for the mapping of E2E reservations

onto aggregate reservations, the same methods MUST be used as the ones described in Section 4 of [RFC4860], augmented by the following rules:

- o) An Aggregator (also PCN-ingress-node in this document) or Deaggregator (also PCN-egress-node and Decision Point in this document) MUST use one or more policies to determine whether a RSVP generic aggregated reservation can be mapped into an ingress-Egress-aggregate. This can be accomplished by using for the different RSVP generic aggregated reservations the same combinations of ingress and egress identifiers, but with a different PHB-ID value (see [RFC4860]) corresponding to the PCN specifications. In particular, the RSVP SESSION object specified in [RFC4860] augmented with the following information:
 - o) the IPv4 DestAddress, IPv6 DestAddress MUST be set to the IPv4 or IPv6 destination addresses, respectively, of the Deaggregator (PCN-egress-node), see [RFC4860]. Note that the PCN-domain is considered as being only one RSVP hop (for Generic aggregated RSVP or E2E RSVP). This means that the next RSVP hop for the Aggregator in the downstream direction is the Deaggregator and the next RSVP hop for the Deaggregator in the upstream direction is the Aggregator.
 - o) PHB-ID (Per Hop Behavior Identification Code) SHOULD be set equal to PCN-compatible Diffserv codepoint(s).
 - o) Extended vDstPort SHOULD be set to the IPv4 or IPv6 destination addresses, of the Aggregator (PCN-ingress-node), see [RFC4860].

2.6. Size of Aggregate Reservations

Since:(i) no admission control of E2 reservations over the RSVP aggregated reservations is required, and (ii) no admission control of the RSVP aggregated reservation over the PCN core is required, the size of the generic aggregate reservation is irrelevant and can be set to any arbitrary value by the Deaggregator. The Deaggregator SHOULD set the value of a generic aggregate reservation to a null bandwidth. We also observe that there is no need for dynamic adjustment of the RSVP aggregated reservation size.

2.7. E2E Path ADSPEC update

To comply with this specification, for the update of the E2E Path ADSPEC, the same methods can be used as the ones described in [RFC4860].

2.8. Intra-domain Routes

The PCN-interior-nodes are neither maintaining E2E RSVP nor RSVP generic aggregation states and reservations. Therefore, intra-domain route changes will not affect intra-domain reservations since such reservations are not maintained by the PCN-interior-nodes.

Furthermore, it is considered that by configuration, the PCN-interior-nodes are not able to distinguish neither RSVP generic aggregated sessions and their associated messages [RFC4860], nor E2E RSVP sessions and their associated messages [RFC2205].

2.9. Inter-domain Routes

The PCN-charter scope precludes inter-domain considerations. However, for solving inter-domain routes changes associated with the operation of the RSVP messages, the same methods SHOULD be used as the ones described in [RFC4860] and in Section 1.4.7 of [RFC3175].

2.10. Reservations for Multicast Sessions

PCN does not consider reservations for multicast sessions.

2.11. Multi-level Aggregation

PCN does not consider multi-level aggregations within the PCN domain. Therefore, the PCN-interior-nodes are not supporting multi-level aggregation procedures. However, the Aggregator and Deaggregator SHOULD support the multi-level aggregation procedures specified in [RFC4860] and in Section 1.4.9 of [RFC3175].

2.12. Reliability Issues

To comply with this specification, for solving possible reliability issues, the same methods MUST be used as the ones described in Section 4 of [RFC4860].

3. Elements of Procedure

This section describes the procedures used to implement the aggregated RSVP procedure over PCN. It is considered that the procedures for aggregation of E2E reservations over generic aggregate RSVP reservations are same as the procedures specified in Section 4 of [RFC4860] except where a departure from these procedures is explicitly described in the present section. Please refer to [RFC4860] for all the below error cases:

- o) Incomplete message
- o) Unexpected objects

3.1. Receipt of E2E Path Message by Aggregating router

When the E2E Path message arrives at the exterior interface of the Aggregator, (also PCN-ingress-node in this document), then standard RSVP generic aggregation [RFC4860] procedures are used.

3.2. Handling Of E2E Path Message by Interior Routers

The E2E Path messages traverse zero or more PCN-interior-nodes. The PCN-interior-nodes receive the E2E Path message on an interior interface and forward it on another interior interface. It is considered that, by configuration, the PCN-interior-nodes ignore the E2E RSVP signaling messages [RFC2205]. Therefore, the E2E Path messages are simply forwarded as normal IP datagrams.

3.3. Receipt of E2E Path Message by Deaggregating router

When receiving the E2E Path message the Deaggregator (also PCN-egress-node and Decision Point in this document) performs the regular [RFC4860] procedures, augmented with the following rules:

- o) The Deaggregator MUST NOT perform the RSVP-TTL vs IP TTL-check and MUST NOT update the ADspec Break bit. This is because the whole PCN-domain is effectively handled by E2E RSVP as a virtual link on which integrated service is indeed supported (and admission control performed) so that the Break bit MUST NOT be set, see also [draft-lefaucheur-rsvp-ecn-01].

The Deaggregator forwards the E2E Path message towards the receiver.

3.4. Initiation of new Aggregate Path Message by Aggregating Router

To comply with this specification, for the initiation of the new RSVP generic aggregated Path message by the Aggregator (also PCN-ingress-node in this document), the same methods MUST be used as the ones described in [RFC4860].

3.5. Handling Of Aggregate Path Message By Interior Routers

The Aggregate Path messages traverse zero or more PCN-interior-nodes. The PCN-interior-nodes receive the Aggregated Path message on an interior interface and forward it on another interior interface. It is considered that, by configuration, the PCN-interior-nodes ignore the Aggregated Path signaling messages. Therefore, the Aggregated Path messages are simply forwarded as normal IP datagrams.

3.6. Handling Of Aggregate Path Message By Deaggregating Router

When receiving the Aggregated Path message, the Deaggregator (also PCN-egress-node and Decision Point in this document) performs the regular [RFC4860] procedures, augmented with the following rules:

- o) When the received Aggregated Path message by the Deaggregator contains the RSVP-AGGREGATE-IPv4-PCN-response or RSVP-AGGREGATE-IPv6-PCN-response PCN objects, which carry the PCN-sent-rate, then the procedures specified in Section 3.18 of this document MUST be followed.

3.7. Handling of E2E Resv Message by Deaggregating Router

When the E2E Resv message arrives at the exterior interface of the Deaggregator, (also PCN-egress-node and Decision Point in this document) then standard RSVP aggregation [RFC4860] procedures are used, augmented with the following rules:

- o) The E2E RSVP session associated with an E2E Resv message that arrives at the external interface of the Deaggregator is mapped/matched with an RSVP generic aggregate and with a PCN ingress-egress-aggregate.
- o) Depending on the type of the PCN edge behavior supported by the Deaggregator, the PCN admission control procedures specified in Section 3.3.1 of [RFC6661] or [RFC6662] MUST be followed. Since no admission control procedures over the RSVP aggregated reservations in the PCN-core are required, unlike [RFC4860], the Deaggregator does not perform any admission control of the E2E Reservation over the mapped generic aggregate RSVP reservation. If the PCN based admission control procedure is successful then the Deaggregator MUST allow the new flow to be admitted onto the associated RSVP generic aggregation reservation and onto the PCN ingress-egress-aggregate, see [RFC6661] and [RFC6662]. If the PCN based admission control procedure is not successful, then the E2E Resv MUST NOT be admitted onto the associated RSVP generic aggregate reservation and onto the PCN ingress-egress-aggregation. The E2E Resv message is further processed according to [RFC4860].

The way of how the PCN-admission-state is maintained is specified in [RFC6661] and [RFC6662].

3.8. Handling Of E2E Resv Message By Interior Routers

The E2E Resv messages traversing the PCN core are IP addressed to the Aggregating router and are not marked with Router Alert, therefore the E2E Resv messages are simply forwarded as normal IP datagrams.

3.9. Initiation of New Aggregate Resv Message By Deaggregating Router

To comply with this specification, for the initiation of the new RSVP generic aggregated Resv message by the Deaggregator (also PCN-egress-node and Decision Point in this document), the same methods MUST be used as the ones described in Section 4 of [RFC4860] augmented with the following rules:

- o) The size of the generic aggregate reservation is irrelevant, see Section 2.6, and can be set to any arbitrary value by the PCN-egress node. The Deaggregator SHOULD set the value of a RSVP generic aggregate reservation to a null bandwidth. We also observe that there is no need for dynamic adjustment of the RSVP generic aggregated reservation size.

- o) When [RFC6661] is used and the ETM-rate measured by the Deaggregator contains a non-zero value for some ingress-egress-aggregate, see [RFC6661] and [RFC6662], the Deaggregator MUST request the PCN-ingress-node to provide an estimate of the rate (PCN-sent-rate) at which the Aggregator (also PCN-ingress-node in this document) is receiving PCN-traffic that is destined for the given ingress-egress-aggregate.
- o) When [RFC6662] is used and the PCN-admission-state computed by the Deaggregator, on the basis of the CLE is "block" for the given ingress-egress-aggregate, the Deaggregator MUST request the PCN-ingress-node to provide an estimate of the rate (PCN-sent-rate) at which the Aggregator is receiving PCN-traffic that is destined for the given ingress-egress-aggregate.
- o) In the above two cases and when the PCN-sent-rate needs to be requested from the Aggregator, the Deaggregator MUST generate and send an (refresh) Aggregated Resv message to the Aggregator that MUST carry one of the following PCN objects, see Section 4.1, depending on whether IPv4 or IPv6 is supported:
 - o) RSVP-AGGREGATE-IPv4-PCN-request
 - o) RSVP-AGGREGATE-IPv6-PCN-request.

3.10. Handling of Aggregate Resv Message by Interior Routers

The Aggregated Resv messages traversing the PCN core are IP addressed to the Aggregating router and are not marked with Router Alert, therefore the Aggregated Resv messages are simply forwarded as normal IP datagrams.

3.11. Handling of E2E Resv Message by Aggregating Router

When the E2E Resv message arrives at the interior interface of the Aggregator (also PCN-ingress-node in this document), then standard RSVP aggregation [RFC4860] procedures are used.

3.12. Handling of Aggregated Resv Message by Aggregating Router

When the Aggregated Resv message arrives at the interior interface of the Aggregator, (also PCN-ingress-node in this document), then standard RSVP aggregation [RFC4860] procedures are used, augmented with the following rules:

- o) the Aggregator SHOULD use the information carried by the PCN objects, see Section 4, and follow the steps specified in [RFC6661], [RFC6662]. If the "R" flag carried by the RSVP-AGGREGATE-IPv4-PCN-request or RSVP-AGGREGATE-IPv6-PCN-request PCN objects is set to ON, see Section 4.1, then the Aggregator follows the steps described in Section 3.4 of [RFC6661] and [RFC6662] on calculating the PCN-sent-rate. In particular, the Aggregator MUST provide the estimated current rate of PCN-traffic received at that node and destined for a given ingress-egress-aggregate in octets per second (the PCN-sent-rate). The way this rate estimate is derived is a matter of implementation, see [RFC6661] or [RFC6662].

- o) the Aggregator initiates an Aggregated Path message. In particular, when the Aggregator receives an Aggregated Resv message which carries one of the following PCN objects: RSVP-AGGREGATE-IPv4-PCN-request or RSVP-AGGREGATE-IPv6-PCN-request, with the flag "R" set to ON, see Section 4.1, the Aggregator initiates an Aggregated Path message, and includes the calculated PCN-sent-rate into the RSVP-AGGREGATE-IPv4-PCN-response or RSVP-AGGREGATE-IPv6-PCN-response PCN objects, see Section 4.1, which that MUST be carried by the Aggregated Path message. This Aggregated Path message is sent towards the Deaggregator (also PCN-egress-node and Decision Point in this document) that requested the calculation of the PCN-sent-rate.

3.13. Removal of E2E Reservation

To comply with this specification, for the removal of E2E reservations, the same methods MUST be used as the ones described in Section 4 of [RFC4860] and [RFC4495].

3.14. Removal of Aggregate Reservation

To comply with this specification, for the removal of RSVP generic aggregated reservations, the same methods MUST be used as the ones described in Section 4 of [RFC4860] and Section 2.10 of [RFC3175]. In particular, should an aggregate reservation go away (presumably due to a configuration change, route change, or policy event), the E2E reservations it supports are no longer active. They MUST be treated accordingly.

3.15. Handling of Data On Reserved E2E Flow by Aggregating Router

The handling of data on the reserved E2E flow by Aggregator (also PCN-ingress-node in this document) uses the procedures described in [RFC4860] augmented with:

- o) Regarding, PCN marking and traffic classification the procedures defined in Section 2.2 and 2.3 of this document are used.

3.16. Procedures for Multicast Sessions

In this document no multicast sessions are considered.

3.17. Misconfiguration of PCN-node

In an event where a PCN-node is misconfigured within a PCN-domain, the desired behavior is same as described in Section 3.10.

3.18 PCN based Flow Termination

When the Deaggregator (also PCN-egress-node and Decision Point in this document) needs to terminate an amount of traffic associated with one ingress-egress-aggregate (see Section 3.3.2 of [RFC6661] and [RFC6662]), then several procedures of terminating E2E microflows can be deployed. The default procedure of terminating E2E microflows (i.e., PCN-flows) is as follows, see i.e., [RFC6661] and [RFC6662].

For the same ingress-egress-aggregate, select a number of E2E microflows to be terminated in order to decrease the total incoming amount of bandwidth associated with one ingress-egress-aggregate by the amount of traffic to be terminated, see above. In this situation the same mechanisms for terminating an E2E microflow can be followed as specified in [RFC2205]. However, based on a local policy, the Deaggregator could use other ways of selecting which microflows should be terminated. For example, for the same ingress-egress-aggregate, select a number of E2E microflows to be terminated or to reduce their reserved bandwidth in order to decrease the total incoming amount of bandwidth associated with one ingress-egress-aggregate by the amount of traffic to be terminated. In this situation the same mechanisms for terminating an E2E microflow or reducing bandwidth associated with an E2E microflow can be followed as specified in [RFC4495].

4. Protocol Elements

The protocol elements in this document are using the ones defined in Section 4 of [RFC4860] and Section 3 of [RFC3175] augmented with the following rules:

- o) the DSCP value included in the SESSION object, SHOULD be set equal to a PCN-compatible Diffserv codepoint.
- o) Extended vDstPort SHOULD be set to the IPv4 or IPv6 destination addresses, of the Aggregator (also PCN-ingress-node in this document), see [RFC4860].
- o) When the Deaggregator (also PCN-egress-node and Decision Point in this document) needs to request the PCN-sent-rate from the PCN-ingress-node, see Section 3.9 of this document, the Deaggregator MUST generate and send an (refresh) Aggregate Resv message to the Aggregator that MUST carry one of the following PCN objects, see Section 4.1, depending on whether IPv4 or IPv6 is supported:
 - o) RSVP-AGGREGATE-IPv4-PCN-request
 - o) RSVP-AGGREGATE-IPv6-PCN-request.
- o) When the Aggregator receives an Aggregate Resv message which carries one of the following PCN objects:
RSVP-AGGREGATE-IPv4-PCN-request or
RSVP-AGGREGATE-IPv6-PCN-request, with the flag "R" set to ON, see Section 4.1, then the Aggregator MUST generate and send to the Deaggregator an Aggregated Path message which carries one of the following PCN objects, see Section 4.1, depending on whether IPv4 or IPv6 is supported:
 - o) RSVP-AGGREGATE-IPv4-PCN-response,
 - o) RSVP-AGGREGATE-IPv6-PCN-response.

4.1 PCN objects

This section describes four types of PCN objects that can be carried by the (refresh) Aggregate Path or the (refresh) Aggregate Resv messages specified in [RFC4860].

These objects are:

- o RSVP-AGGREGATE-IPv4-PCN-request,
- o RSVP-AGGREGATE-IPv6-PCN-request,
- o RSVP-AGGREGATE-IPv4-PCN-response,
- o RSVP-AGGREGATE-IPv6-PCN-response.

- o) RSVP-AGGREGATE-IPv4-PCN-request: PCN request object, when IPv4 addresses are used:

Class = 248 (PCN)

C-Type = 1 (RSVP-AGGREGATE-IPv4-PCN-request)

+-----+-----+-----+-----+	
	IPv4 PCN-ingress-node Address (4 bytes)
+-----+-----+-----+-----+	
	IPv4 PCN-egress-node Address (4 bytes)
+-----+-----+-----+-----+	
	IPv4 Decision Point Address (4 bytes)
+-----+-----+-----+-----+	
R	Reserved
+-----+-----+-----+-----+	

- o) RSVP-AGGREGATE-IPv6-PCN-request: PCN object, when IPv6 addresses are used:

Class = 248 (PCN)

C-Type = 2 (RSVP-AGGREGATE-IPv6-PCN-request)

+-----+-----+-----+-----+	
	IPv6 PCN-ingress-node Address (16 bytes)
+	
+	
+-----+-----+-----+-----+	
	IPv6 PCN-egress-node Address (16 bytes)
+	
+	
+-----+-----+-----+-----+	
	Decision Point Address (16 bytes)
+	
+	
+-----+-----+-----+-----+	
R	Reserved
+-----+-----+-----+-----+	

- o) RSVP-AGGREGATE-IPv4-PCN-response: PCN object, IPv4 addresses are used:
 Class = 248 (PCN)
 C-Type = 3 (RSVP-AGGREGATE-IPv4-PCN-response)

```

+-----+-----+-----+-----+
| IPv4 PCN-ingress-node Address (4 bytes) |
+-----+-----+-----+-----+
| IPv4 PCN-egress-node Address (4 bytes) |
+-----+-----+-----+-----+
| IPv4 Decision Point Address (4 bytes) |
+-----+-----+-----+-----+
| PCN-sent-rate |
+-----+-----+-----+-----+

```

- o) RSVP-AGGREGATE-IPv6-PCN-response: PCN object, IPv6 addresses are used:
 Class = 248 (PCN)
 C-Type = 4 (RSVP-AGGREGATE-IPv6-PCN-response)

```

+-----+-----+-----+-----+
|                                     |
+                                     +
| IPv6 PCN-ingress-node Address (16 bytes) |
+                                     +
|                                     |
+-----+-----+-----+-----+
|                                     |
+                                     +
| IPv6 PCN-egress-node Address (16 bytes) |
+                                     +
|                                     |
+-----+-----+-----+-----+
|                                     |
+                                     +
| Decision Point Address (16 bytes) |
+                                     +
|                                     |
+-----+-----+-----+-----+
| PCN-sent-rate |
+-----+-----+-----+-----+

```

The fields carried by the PCN object are specified in [RFC6663], [RFC6661] and [RFC6662]:

- o the IPv4 or IPv6 address of the PCN-ingress-node (Aggregator) and the IPv4 or IPv6 address of the PCN-egress-node (Deaggregator); together they specify the ingress-egress-aggregate to which the report refers. According to [RFC6663] the report should carry the identifier of the PCN-ingress-node (Aggregator) and the identifier of the PCN-egress-node (Deaggregator) (typically their IP addresses);
- o Decision Point address specify the IPv4 or IPv6 address of the Decision Point. In this document this field MUST contain the IP address of the Deaggregator.
- o "R": 1 bit flag that when set to ON, signifies, according to [RFC6661] and [RFC6662], that the PCN-ingress-node (Aggregator) MUST provide an estimate of the rate (PCN-sent-rate) at which the PCN-ingress-node (Aggregator) is receiving PCN-traffic that is destined for the given ingress-egress-aggregate.
- o "Reserved": 31 bits that are currently not used by this document and are reserved. These SHALL be set to 0 and SHALL be ignored on reception.
- o PCN-sent-rate: the PCN-sent-rate for the given ingress-egress-aggregate. It is expressed in octets/second; its format is a 32-bit IEEE floating point number; The PCN-sent-rate is specified in [RFC6661] and [RFC6662] and it represents the estimate of the rate at which the PCN-ingress-node (Aggregator) is receiving PCN-traffic that is destined for the given ingress-egress-aggregate.

5. Security Considerations

The security considerations specified in [RFC2205], [RFC4860] and [RFC5559] apply to this document. In addition, [RFC4230] and [RFC6411] provide useful guidance on RSVP security mechanisms.

Security within a PCN domain is fundamentally based on the controlled environment trust assumption stated in Section 6.3.1 of [RFC5559], in particular that all PCN-nodes are PCN-enabled and are trusted to perform accurate PCN-metering and PCN-marking.

In the PCN domain environments addressed by this document, Generic Aggregate Resource ReSerVation Protocol (RSVP) messages specified in [RFC4860] are used for support of the PCN Controlled Load (CL) and Single Marking (SM) edge behaviors over a Diffserv cloud using Pre-Congestion Notification. Hence the security mechanisms discussed in [RFC4860] are applicable. Specifically, the INTEGRITY object [RFC2747][RFC3097] can be used to provide hop-by-hop RSVP message integrity, node authentication and replay protection, thereby protecting against corruption and spoofing of RSVP messages and PCN feedback conveyed by RSVP messages.

For these reasons, this document does not introduce significant additional security considerations beyond those discussed in

[RFC5559] and [RFC4860].

6. IANA Considerations

IANA has modified the RSVP parameters registry, 'Class Names, Class Numbers, and Class Types' subregistry, to add a new Class Number and assign 4 new C-Types under this new Class Number, as described below, see Section 4.1:

Class Number	Class Name	Reference
-----	-----	-----
248	PCN	this document
Class Types or C-Types:		
1	RSVP-AGGREGATE-IPv4-PCN-request	this document
2	RSVP-AGGREGATE-IPv6-PCN-request	this document
3	RSVP-AGGREGATE-IPv4-PCN-response	this document
4	RSVP-AGGREGATE-IPv6-PCN-response	this document

When this draft is published as an RFC, IANA should update the reference for the above 5 items to that published RFC (and the RFC Editor should remove this sentence).

7. Acknowledgments

We would like to thank the authors of [draft-lefaucheur-rsvp-ecn-01.txt], since some ideas used in this document are based on the work initiated in [draft-lefaucheur-rsvp-ecn-01.txt]. Moreover, we would like to thank Bob Briscoe, David Black, Ken Carlberg, Tom Taylor, Philip Eardley, Michael Menth, Toby Moncaster, James Polk, Scott Bradner, Lixia Zhang and Robert Sparks for the provided comments. In particular, we would like to thank Francois Le Faucheur for contributing in addition to comments also to a significant amount of text.

8. Normative References

[RFC6661] T. Taylor, A. Charny, F. Huang, G. Karagiannis, M. Menth, "PCN Boundary Node Behaviour for the Controlled Load (CL) Mode of Operation", July 2012.

[RFC6662] A. Charny, J. Zhang, G. Karagiannis, M. Menth, T. Taylor, "PCN Boundary Node Behaviour for the Single Marking (SM) Mode of Operation", July 2012.

[RFC6663] G. Karagiannis, T. Taylor, K. Chan, M. Menth, P. Eardley, "Requirements for Signaling of (Pre-) Congestion Information in a DiffServ Domain", July 2012.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2205] Braden, R., ed., et al., "Resource ReSerVation Protocol (RSVP)- Functional Specification", RFC 2205, September 1997.
- [RFC3140] Black, D., Brim, S., Carpenter, B., and F. Le Faucheur, "Per Hop Behavior Identification Codes", RFC 3140, June 2001.
- [RFC3175] Baker, F., Iturralde, C., Le Faucheur, F., and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3175, September 2001.
- [RFC4495] Polk, J. and S. Dhesikan, "A Resource Reservation Protocol (RSVP) Extension for the Reduction of Bandwidth of a Reservation Flow", RFC 4495, May 2006.
- [RFC4860] F. Le Faucheur, B. Davie, P. Bose, C. Christou, M. Davenport, "Generic Aggregate Resource ReSerVation Protocol (RSVP) Reservations", RFC4860, May 2007.
- [RFC5670] Eardley, P., "Metering and Marking Behaviour of PCN-Nodes", RFC 5670, November 2009.
- [RFC6660] Moncaster, T., Briscoe, B., and M. Menth, "Baseline Encoding and Transport of Pre-Congestion Information", RFC 6660, July 2012.

9. Informative References

- [draft-lefaucheur-rsvp-ecn-01.txt] Le Faucheur, F., Charny, A., Briscoe, B., Eardley, P., Chan, K., and J. Babiarz, "RSVP Extensions for Admission Control over Diffserv using Pre-congestion Notification (PCN) (Work in progress)", June 2006.
- [RFC1633] Braden, R., Clark, D., and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.
- [RFC2211] J. Wroclawski, Specification of the Controlled-Load Network Element Service, September 1997
- [RFC2212] S. Shenker et al., Specification of Guaranteed Quality of Service, September 1997
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "A framework for Differentiated Services", RFC 2475, December 1998.

[RFC2747] Baker, F., Lindell, B., and M. Talwar, "RSVP Cryptographic Authentication", RFC 2747, January 2000.

[RFC2753] Yavatkar, R., D. Pendarakis and R. Guerin, "A Framework for Policy-based Admission Control", January 2000.

[RFC2998] Bernet, Y., Yavatkar, R., Ford, P., Baker, F., Zhang, L., Speer, M., Braden, R., Davie, B., Wroclawski, J. and E. Felstaine, "A Framework for Integrated Services Operation Over DiffServ Networks", RFC 2998, November 2000.

[RFC3097] Braden, R. and L. Zhang, "RSVP Cryptographic Authentication -- Updated Message Type Value", RFC 3097, April 2001.

[RFC4230] H. Tschofenig, R. Graveman, "RSVP Security Properties", RFC 4230, December 2005.

[RFC5559] Eardley, P., "Pre-Congestion Notification (PCN) Architecture", RFC 5559, June 2009.

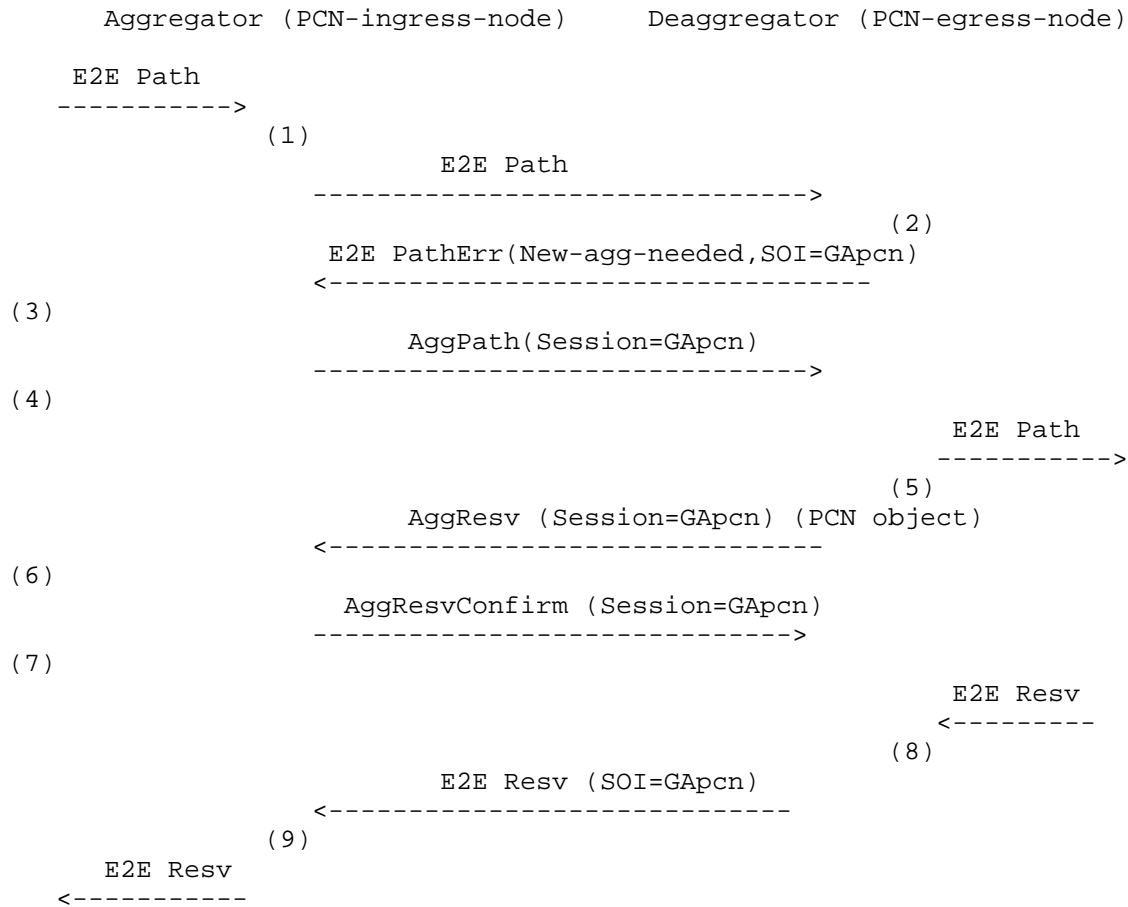
[RFC6411] M. Behringer, F. Le Faucheur, B. Weis, "Applicability of Keying Methods for RSVP Security", RFC 6411, October 2011.

[SIG-NESTED] Baker, F. and P. Bose, "QoS Signaling in a Nested Virtual Private Network", Work in Progress, July 2007.

10. Appendix A: Example Signaling Flow

This appendix is based on the appendix provided in [RFC4860]. In particular, it provides an example signaling flow of the specification detailed in Section 3 and 4.

This signaling flow assumes an environment where E2E reservations are aggregated over generic aggregate RSVP reservations and applied over a PCN domain. In particular the Aggregator (PCN-ingress-node) and Deaggregator (PCN-egress-node) are located at the boundaries of the PCN domain. The PCN-interior-nodes are located within the PCN-domain, between the PCN-boundary nodes, but are not shown in this Figure. It illustrates a possible RSVP message flow that could take place in the successful establishment of a unicast E2E reservation that is the first between a given pair of Aggregator/Deaggregator.



(1) The Aggregator forwards E2E Path into the aggregation region after modifying its IP protocol number to RSVP-E2E-IGNORE

(2) Let's assume no Aggregate Path exists. To be able to accurately update the ADSPEC of the E2E Path, the Deaggregator needs the ADSPEC of Aggregate Path. In this example, the Deaggregator elects to instruct the Aggregator to set up an Aggregate Path state for the PCN PHB-ID. To do that, the Deaggregator sends an E2E PathErr message with a New-Agg-Needed PathErr code.

The PathErr message also contains a SESSION-OF-INTEREST (SOI) object. The SOI contains a GENERIC-AGGREGATE SESSION (GApcn) whose PHB-ID is set to the PCN PHB-ID. The GENERIC-AGGREGATE SESSION contains an interface-independent Deaggregator address inside the DestAddress and appropriate values inside the vDstPort and Extended vDstPort fields. In this document, the Extended vDstPort SHOULD contain the IPv4 or IPv6 address of the Aggregator.

(3) The Aggregator follows the request from the Deaggregator and

signals an Aggregate Path for the GENERIC-AGGREGATE Session (GApcn).

- (4) The Deaggregator takes into account the information contained in the ADSPEC from both Aggregate Paths and updates the E2E Path ADSPEC accordingly. The PCN-egress-node MUST NOT perform the RSVP-TTL vs IP TTL-check and MUST NOT update the ADSpec Break bit. This is because the whole PCN-domain is effectively handled by E2E RSVP as a virtual link on which integrated service is indeed supported (and admission control performed) so that the Break bit MUST NOT be set, see also [draft-lefaucheur-rsvp-ecn-01]. The Deaggregator also modifies the E2E Path IP protocol number to RSVP before forwarding it.
- (5) In this example, the Deaggregator elects to immediately proceed with establishment of the generic aggregate reservation. In effect, the Deaggregator can be seen as anticipating the actual demand of E2E reservations so that the generic aggregate reservation is in place when the E2E Resv request arrives, in order to speed up establishment of E2E reservations. Here it is also assumed that the Deaggregator includes the optional Resv Confirm Request in the Aggregate Resv message.
- (6) The Aggregator merely complies with the received ResvConfirm Request and returns the corresponding Aggregate ResvConfirm.
- (7) The Deaggregator has explicit confirmation that the generic aggregate reservation is established.
- (8) On receipt of the E2E Resv, the Deaggregator applies the mapping policy defined by the network administrator to map the E2E Resv onto a generic aggregate reservation. Let's assume that this policy is such that the E2E reservation is to be mapped onto the generic aggregate reservation with the PCN PHB-ID=x. The Deaggregator knows that a generic aggregate reservation (GApcn) is in place for the corresponding PHB-ID since (7). At this step the Deaggregator maps the generic aggregated reservation onto one ingress-egress-aggregate maintained by the Deaggregator (as a PCN-egress-node), see Section 3.7. The Deaggregator performs admission control of the E2E Resv onto the generic Aggregate reservation for the PCN PHB-ID (GApcn). The Deaggregator takes also into account the PCN admission control procedure as specified in [RFC6661] and [RFC6662], see Section 3.7. If one or both the admission control procedures (PCN based admission control procedure and admission control procedure specified in [RFC4860]) are not successful, then the E2E Resv is not admitted onto the associated RSVP generic aggregate reservation for the PCN PHB-ID (GApcn). Otherwise, assuming that the generic aggregate reservation for the PCN (GApcn) had been established with sufficient bandwidth to support the E2E Resv, the Deaggregator adjusts its counter, tracking the unused bandwidth on the generic aggregate reservation. Then it forwards the E2E Resv to the Aggregator including a SESSION-OF-INTEREST

object conveying the selected mapping onto GApcn (and hence onto the PCN PHB-ID).

- (9) The Aggregator records the mapping of the E2E Resv onto GApcn (and onto the PCN PHB-ID). The Aggregator removes the SOI object and forwards the E2E Resv towards the sender.

11. Authors' Address

Georgios Karagiannis
Huawei Technologies
Hansaallee 205,
40549 Dusseldorf,
Germany
Email: Georgios.Karagiannis@huawei.com

Anurag Bhargava
Cisco Systems, Inc.
7100-9 Kit Creek Road
PO Box 14987
RESEARCH TRIANGLE PARK, NORTH CAROLINA 27709-4987
USA
Email: anuragb@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 28, 2015

M. Tuexen
Muenster Univ. of Appl. Sciences
R. Stewart
Netflix, Inc.
R. Jesup
WorldGate Communications
S. Loreto
Ericsson
January 24, 2015

DTLS Encapsulation of SCTP Packets
draft-ietf-tsvwg-sctp-dtls-encaps-09.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a transport protocol originally defined to run on top of the network protocols IPv4 or IPv6. This document specifies how SCTP can be used on top of the Datagram Transport Layer Security (DTLS) protocol. Using the encapsulation method described in this document, SCTP is unaware of the protocols being used below DTLS; hence explicit IP addresses cannot be used in the SCTP control chunks. As a consequence, the SCTP associations carried over DTLS can only be single homed.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 28, 2015.

Copyright Notice

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

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

Table of Contents

1. Overview	2
2. Conventions	3
3. Encapsulation and Decapsulation Procedure	3
4. General Considerations	3
5. DTLS Considerations	4
6. SCTP Considerations	5
7. IANA Considerations	6
8. Security Considerations	6
9. Acknowledgments	7
10. References	7
Appendix A. NOTE to the RFC-Editor	9
Authors' Addresses	9

1. Overview

The Stream Control Transmission Protocol (SCTP) as defined in [RFC4960] is a transport protocol running on top of the network protocols IPv4 [RFC0791] or IPv6 [RFC2460]. This document specifies how SCTP is used on top of the Datagram Transport Layer Security (DTLS) protocol. DTLS 1.0 is defined in [RFC4347] and the latest version when this RFC was published, DTLS 1.2, is defined in [RFC6347]. This encapsulation is used for example within the WebRTC protocol suite (see [I-D.ietf-rtcweb-overview] for an overview) for transporting non-SRTP data between browsers. The architecture of this stack is described in [I-D.ietf-rtcweb-data-channel].

[NOTE to RFC-Editor:

Please ensure that the authors double check the above statement about DTLS 1.2 during AUTH48 and then remove this note before publication.

]

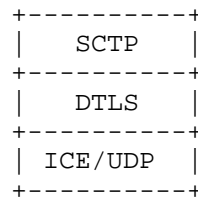


Figure 1: Basic stack diagram

This encapsulation of SCTP over DTLS over UDP or ICE/UDP (see [RFC5245]) can provide a NAT traversal solution in addition to confidentiality, source authentication, and integrity protected transfers. Please note that using ICE does not necessarily imply that a different packet format is used on the wire.

Please note that the procedures defined in [RFC6951] for dealing with the UDP port numbers do not apply here. When using the encapsulation defined in this document, SCTP is unaware about the protocols used below DTLS.

2. Conventions

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

3. Encapsulation and Decapsulation Procedure

When an SCTP packet is provided to the DTLS layer, the complete SCTP packet, consisting of the SCTP common header and a number of SCTP chunks, is handled as the payload of the application layer protocol of DTLS. When the DTLS layer has processed a DTLS record containing a message of the application layer protocol, the payload is passed to the SCTP layer. The SCTP layer expects an SCTP common header followed by a number of SCTP chunks.

4. General Considerations

An implementation of SCTP over DTLS MUST implement and use a path maximum transmission unit (MTU) discovery method that functions without ICMP to provide SCTP/DTLS with an MTU estimate. An implementation of "Packetization Layer Path MTU Discovery" [RFC4821] either in SCTP or DTLS is RECOMMENDED.

The path MTU discovery is performed by SCTP when SCTP over DTLS is used for data channels (see Section 5 of [I-D.ietf-rtcweb-data-channel]).

5. DTLS Considerations

The DTLS implementation MUST support DTLS 1.0 [RFC4347] and SHOULD support the most recently published version of DTLS, which was DTLS 1.2 [RFC6347] when this RFC was published. In the absence of a revision to this document, the latter requirement applies to all future versions of DTLS when they are published as RFCs. This document will only be revised if a revision to DTLS or SCTP makes a revision to the encapsulation necessary.

[NOTE to RFC-Editor:

Please ensure that the authors double check the above statement about DTLS 1.2 during AUTH48 and then remove this note before publication.

]

SCTP performs segmentation and reassembly based on the path MTU. Therefore the DTLS layer MUST NOT use any compression algorithm.

The DTLS MUST support sending messages larger than the current path MTU. This might result in sending IP level fragmented messages.

If path MTU discovery is performed by the DTLS layer, the method described in [RFC4821] MUST be used. For probe packets, the extension defined in [RFC6520] MUST be used.

If path MTU discovery is performed by the SCTP layer and IPv4 is used as the network layer protocol, the DTLS implementation SHOULD allow the DTLS user to enforce that the corresponding IPv4 packet is sent with the Don't Fragment (DF) bit set. If controlling the DF bit is not possible, for example due to implementation restrictions, a safe value for the path MTU has to be used by the SCTP stack. It is RECOMMENDED that the safe value does not exceed 1200 bytes. Please note that [RFC1122] only requires end hosts to be able to reassemble fragmented IP packets up to 576 bytes in length.

The DTLS implementation SHOULD allow the DTLS user to set the Differentiated services code point (DSCP) used for IP packets being sent (see [RFC2474]). This requires the DTLS implementation to pass the value through and the lower layer to allow setting this value. If the lower layer does not support setting the DSCP, then the DTLS user will end up with the default value used by protocol stack. Please note that only a single DSCP value can be used for all packets belonging to the same SCTP association.

Using explicit congestion notifications (ECN) in SCTP requires the DTLS layer to pass the ECN bits through and its lower layer to expose access to them for sent and received packets (see [RFC3168]). The implementation of DTLS and its lower layer have to provide this support. If this is not possible, for example due to implementation restrictions, ECN can't be used by SCTP.

6. SCTP Considerations

This section describes the usage of the base protocol and the applicability of various SCTP extensions.

6.1. Base Protocol

This document uses SCTP [RFC4960] with the following restrictions, which are required to reflect that the lower layer is DTLS instead of IPv4 and IPv6 and that SCTP does not deal with the IP addresses or the transport protocol used below DTLS:

- o A DTLS connection MUST be established before an SCTP association can be set up.
- o Multiple SCTP associations MAY be multiplexed over a single DTLS connection. The SCTP port numbers are used for multiplexing and demultiplexing the SCTP associations carried over a single DTLS connection.
- o All SCTP associations are single-homed, because DTLS does not expose any address management to its upper layer. Therefore it is RECOMMENDED to set the SCTP parameter `path.max.retrans` to `association.max.retrans`.
- o The INIT and INIT-ACK chunk MUST NOT contain any IPv4 Address or IPv6 Address parameters. The INIT chunk MUST NOT contain the Supported Address Types parameter.
- o The implementation MUST NOT rely on processing ICMP or ICMPv6 packets, since the SCTP layer most likely is unable to access the SCTP common header in the plain text of the packet, which triggered the sending of the ICMP or ICMPv6 packet. This applies in particular to path MTU discovery when performed by SCTP.
- o If the SCTP layer is notified about a path change by its lower layers, SCTP SHOULD retest the Path MTU and reset the congestion state to the initial state. The window-based congestion control method specified in [RFC4960], resets the congestion window and slow start threshold to their initial values.

6.2. Padding Extension

When the SCTP layer performs path MTU discovery as specified in [RFC4821], the padding extension defined in [RFC4820] MUST be supported and used for probe packets (HEARTBEAT chunks bundled with PADDING chunks [RFC4820]).

6.3. Dynamic Address Reconfiguration Extension

If the dynamic address reconfiguration extension defined in [RFC5061] is used, ASCONF chunks MUST use wildcard addresses only.

6.4. SCTP Authentication Extension

The SCTP authentication extension defined in [RFC4895] can be used with DTLS encapsulation, but does not provide any additional benefit.

6.5. Partial Reliability Extension

Partial reliability as defined in [RFC3758] can be used in combination with DTLS encapsulation. It is also possible to use additional PR-SCTP policies, for example the ones defined in [I-D.ietf-tsvwg-sctp-prpolicies].

6.6. Stream Reset Extension

The SCTP stream reset extension defined in [RFC6525] can be used with DTLS encapsulation. It is used to reset SCTP streams and add SCTP streams during the lifetime of the SCTP association.

6.7. Interleaving of Large User Messages

SCTP as defined in [RFC4960] does not support the interleaving of large user messages that need to be fragmented and reassembled by the SCTP layer. The protocol extension defined in [I-D.ietf-tsvwg-sctp-ndata] overcomes this limitation and can be used with DTLS encapsulation.

7. IANA Considerations

This document requires no actions from IANA.

8. Security Considerations

Security considerations for DTLS are specified in [RFC4347] and for SCTP in [RFC4960], [RFC3758], and [RFC6525]. The combination of SCTP and DTLS introduces no new security considerations.

SCTP should not process the IP addresses used for the underlying communication since DTLS provides no guarantees about them.

It should be noted that the inability to process ICMP or ICMPv6 messages does not add any security issue. When SCTP is carried over a connection-less lower layer like IPv4, IPv6, or UDP, processing of these messages is required to protect other nodes not supporting SCTP. Since DTLS provides a connection-oriented lower layer, this kind of protection is not necessary.

9. Acknowledgments

The authors wish to thank David Black, Benoit Claise, Spencer Dawkins, Francis Dupont, Gorrry Fairhurst, Stephen Farrell, Christer Holmberg, Barry Leiba, Eric Rescorla, Tom Taylor, Joe Touch and Magnus Westerlund for their invaluable comments.

10. References

10.1. Normative References

- [RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.
- [RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, March 2007.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.
- [RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.

10.2. Informative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, August 2007.
- [RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, February 2012.
- [RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013.
- [I-D.ietf-rtcweb-overview] Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-13 (work in progress), November 2014.

[I-D.ietf-rtcweb-data-channel]

Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ietf-tsvwg-sctp-prpolicies]

Tuexen, M., Seggelmann, R., Stewart, R., and S. Loreto, "Additional Policies for the Partial Reliability Extension of the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-prpolicies-06 (work in progress), December 2014.

[I-D.ietf-tsvwg-sctp-ndata]

Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "Stream Schedulers and a New Data Chunk for the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-ndata-02 (work in progress), January 2015.

Appendix A. NOTE to the RFC-Editor

Although the references to [I-D.ietf-tsvwg-sctp-prpolicies] and [I-D.ietf-tsvwg-sctp-ndata] are informative, put this document in REF-HOLD until these two references have been approved and update these references to the corresponding RFCs.

Authors' Addresses

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

Email: tuexen@fh-muenster.de

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

Email: randall@lakerest.net

Randell Jesup
WorldGate Communications
3800 Horizon Blvd, Suite #103
Trevose, PA 19053-4947
US

Phone: +1-215-354-5166
Email: randell_ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: Salvatore.Loreto@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 20, 2016

Y. Nishida
GE Global Research
P. Natarajan
Cisco Systems
A. Caro
BBN Technologies
P. Amer
University of Delaware
K. Nielsen
Ericsson
February 17, 2016

SCTP-PF: Quick Failover Algorithm in SCTP
draft-ietf-tsvwg-sctp-failover-16.txt

Abstract

SCTP supports multi-homing. However, when the failover operation specified in RFC4960 is followed, there can be significant delay and performance degradation in the data transfer path failover. To overcome this problem this document specifies a quick failover algorithm (SCTP-PF) based on the introduction of a Potentially Failed (PF) state in SCTP Path Management.

The document also specifies a dormant state operation of SCTP. This dormant state operation is required to be followed by an SCTP-PF implementation, but it may equally well be applied by a standard RFC4960 SCTP implementation.

Additionally, the document introduces an alternative switchback operation mode called Primary Path Switchover that will be beneficial in certain situations. This mode of operation applies to both a standard RFC4960 SCTP implementation as well as to a SCTP-PF implementation.

The procedures defined in the document require only minimal modifications to the RFC4960 specification. The procedures are sender-side only and do not impact the SCTP receiver.

Status of This Memo

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

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

working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on August 20, 2016.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Conventions and Terminology	4
3. SCTP with Potentially Failed Destination State (SCTP-PF) . .	4
3.1. Overview	4
3.2. Specification of the SCTP-PF Procedures	5
4. Dormant State Operation	9
4.1. SCTP Dormant State Procedure	10
5. Primary Path Switchover	11
6. Suggested SCTP Protocol Parameter Values	12
7. Socket API Considerations	12
7.1. Support for the Potentially Failed Path State	13
7.2. Peer Address Thresholds (SCTP_PEER_ADDR_THLDS) Socket Option	14
7.3. Exposing the Potentially Failed Path State (SCTP_EXPOSE_POTENTIALLY_FAILED_STATE) Socket Option . .	15
8. Security Considerations	15
9. MIB Considerations	16
10. IANA Considerations	16
11. Acknowledgements	16
12. Proposed Change of Status (to be Deleted before Publication)	17
13. References	17

13.1. Normative References	17
13.2. Informative References	17
Appendix A. Discussions of Alternative Approaches	18
A.1. Reduce Path.Max.Retrans (PMR)	18
A.2. Adjust RTO related parameters	19
Appendix B. Discussions for Path Bouncing Effect	20
Appendix C. SCTP-PF for SCTP Single-homed Operation	20
Authors' Addresses	21

1. Introduction

The Stream Control Transmission Protocol (SCTP) specified in [RFC4960] supports multi-homing at the transport layer. SCTP's multi-homing features include failure detection and failover procedures to provide network interface redundancy and improved end-to-end fault tolerance. In SCTP's current failure detection procedure, the sender must experience Path.Max.Retrans (PMR) number of consecutive failed timer-based retransmissions on a destination address before detecting a path failure. Until detecting the path failure, the sender continues to transmit data on the failed path. The prolonged time in which [RFC4960] SCTP continues to use a failed path severely degrades the performance of the protocol. To address this problem, this document specifies a quick failover algorithm (SCTP-PF) based on the introduction of a new Potentially Failed (PF) path state in SCTP path management. The performance deficiencies of the [RFC4960] failover operation, and the improvements obtainable from the introduction of a Potentially Failed state in SCTP, were proposed and documented in [NATARAJAN09] for Concurrent Multipath Transfer SCTP [IYENGAR06].

While SCTP-PF can accelerate failover process and improve performance, the risks that an SCTP endpoint enters the dormant state where all destination addresses are inactive can be increased. [RFC4960] leaves the protocol operation during dormant state to implementations and encourages to avoid entering the state as much as possible by careful tuning of the Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) parameters. We specify a dormant state operation for SCTP-PF which makes SCTP-PF provide the same disruption tolerance as [RFC4960] despite that the dormant state may be entered more quickly. The dormant state operation may equally well be applied by an [RFC4960] implementation and will here serve to provide added fault tolerance for situations where the tuning of the Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) parameters fail to provide adequate prevention of the entering of the dormant state.

The operation after the recovery of a failed path also impacts the performance of the protocol. With the procedures specified in

[RFC4960] SCTP will, after a failover from the primary path, switch back to use the primary path for data transfer as soon as this path becomes available again. From a performance perspective such a forced switchback of the data transmission path can be suboptimal as the CWND towards the original primary destination address has to be rebuilt once data transfer resumes, [CARO02]. As an optional alternative to the switchback operation of [RFC4960], this document specifies an alternative Primary Path Switchover procedure which avoid such forced switchbacks of the data transfer path. The Primary Path Switchover operation was originally proposed in [CARO02].

While SCTP-PF primarily is motivated by a desire to improve the multi-homed operation, the feature applies also to SCTP single-homed operation. Here the algorithm serves to provide increased failure detection on idle associations, whereas the failover or switchback aspects of the algorithm will not be activated. This is discussed in more detail in Appendix C.

A brief description of the motivation for the introduction of the Potentially Failed state including a discussion of alternative approaches to mitigate the deficiencies of the [RFC4960] failover operation are given in the Appendices. Discussion of path bouncing effects that might be caused by frequent switchovers, are also provided there.

2. Conventions and Terminology

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

3. SCTP with Potentially Failed Destination State (SCTP-PF)

3.1. Overview

To minimize the performance impact during failover, the sender should avoid transmitting data to a failed destination address as early as possible. In the [RFC4960] SCTP path management scheme, the sender stops transmitting data to a destination address only after the destination address is marked inactive. This process takes a significant amount of time as it requires the error counter of the destination address to exceed the Path.Max.Retrans (PMR) threshold. The issue cannot simply be mitigated by lowering of the PMR threshold because this may result in spurious failure detection and unnecessary prevention of the usage of a preferred primary path. Also due to the coupled tuning of the Path.Max.Retrans (PMR) and the Association.Max.Retrans (AMR) parameter values in [RFC4960], lowering

of the PMR threshold may result in lowering of the AMR threshold, which would result in decrease of the fault tolerance of SCTP.

The solution provided in this document is to extend the SCTP path management scheme of [RFC4960] by the addition of the Potentially Failed (PF) state as an intermediate state in between the active and inactive state of a destination address in the [RFC4960] path management scheme, and let the failover of data transfer away from a destination address be driven by the entering of the PF state instead of by the entering of the inactive state. Thereby SCTP may perform quick failover without negatively impacting the overall fault tolerance of [RFC4960] SCTP. At the same time, RTO-based HEARTBEAT probing is initiated towards a destination address once it enters PF state. Thereby SCTP may quickly ascertain whether network connectivity towards the destination address is broken or whether the failover was spurious. In the case where the failover was spurious data transfer may quickly resume towards the original destination address.

The new failure detection algorithm assumes that loss detected by a timeout implies either severe congestion or network connectivity failure. It recommends that by default a destination address is classified as PF at the occurrence of the first timeout.

3.2. Specification of the SCTP-PF Procedures

The SCTP-PF operation is specified as follows:

1. The sender maintains a new tunable SCTP Protocol Parameter called PotentiallyFailed.Max.Retrans (PFMR). The PFMR defines the new intermediate PF threshold on the destination address error counter. When this threshold is exceeded the destination address is classified as PF. The RECOMMENDED value of PFMR is 0. If PFMR is set to be greater than or equal to Path.Max.Retrans (PMR), the resulting PF threshold will be so high that the destination address will reach the inactive state before it can be classified as PF.
2. The error counter of an active destination address is incremented or cleared as specified in [RFC4960]. This means that the error counter of the destination address in active state will be incremented each time the T3-rtx timer expires, or each time a HEARTBEAT chunk is sent when idle and not acknowledged within an RTO. When the value in the destination address error counter exceeds PFMR, the endpoint MUST mark the destination address as in the PF state.

3. A SCTP-PF sender SHOULD NOT send data to destination addresses in PF state when alternative destination addresses in active state are available. Specifically this means that:
 - i When there is outbound data to send and the destination address presently used for data transmission is in PF state, the sender SHOULD choose a destination address in active state, if one exists, and use this destination address for data transmission.
 - ii As specified in [RFC4960] section 6.4.1, when the sender retransmits data that has timed out, it should attempt to pick a new destination address for data retransmission. In this case, the sender SHOULD choose an alternate destination transport address in active state if one exists.
 - iii When there is outbound data to send and the SCTP user explicitly requests to send data to a destination address in PF state, the sender SHOULD send the data to an alternate destination address in active state if one exists.

When choosing among multiple destination addresses in active state an SCTP sender will follow the guiding principles of section 6.4.1 of [RFC4960] of choosing most divergent source-destination pairs compared with, for i.: the destination address in PF state that it performs a failover from, and for ii.: the destination address towards which the data timed out. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document.

In all cases, the sender MUST NOT change the state of chosen destination address, whether this state be active or PF, and it MUST NOT clear the error counter of the destination address as a result of choosing the destination address for data transmission.

4. When the destination addresses are all in PF state or some in PF state and some in inactive state, the sender MUST choose one destination address in PF state and SHOULD transmit or retransmit data to this destination address using the following rules:
 - A. The sender SHOULD choose the destination in PF state with the lowest error count (fewest consecutive timeouts) for data transmission and transmit or retransmit data to this destination.

- B. When there are multiple destination addresses in PF state with same error count, the sender should let the choice among the multiple destination addresses in PF state with equal error count be based on the [RFC4960], section 6.4.1, principles of choosing most divergent source-destination pairs when executing (potentially consecutive) retransmission. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document.

The sender MUST NOT change the state and the error counter of any destination addresses as the result of the selection.

- 5. The HB.interval of the Path Heartbeat function of [RFC4960] MUST be ignored for destination addresses in PF state. Instead HEARTBEAT chunks are sent to destination addresses in PF state once per RTO. HEARTBEAT chunks SHOULD be sent to destination addresses in PF state, but the sending of HEARTBEATS MUST honor whether the Path Heartbeat function (Section 8.3 of [RFC4960]) is enabled for the destination address or not. I.e., if the Path Heartbeat function is disabled for the destination address in question, HEARTBEATS MUST NOT be sent. Note that when Heartbeat function is disabled, it may take longer to transition a destination address in PF state back to active state.
- 6. HEARTBEATS are sent when a destination address reaches the PF state. When a HEARTBEAT chunk is not acknowledged within the RTO, the sender increments the error counter and exponentially backs off the RTO value. If the error counter is less than PMR, the sender transmits another packet containing the HEARTBEAT chunk immediately after timeout expiration on the previous HEARTBEAT. When data is being transmitted to a destination address in the PF state, the transmission of a HEARTBEAT chunk MAY be omitted in case where the receipt of a SACK of the data or a T3-rtx timer expiration on the data can provide equivalent information, such as the case where the data chunk has been transmitted to a single destination address only. Likewise, the timeout of a HEARTBEAT chunk MAY be ignored if data is outstanding towards the destination address.
- 7. When the sender receives a HEARTBEAT ACK from a HEARTBEAT sent to a destination address in PF state, the sender SHOULD clear the error counter of the destination address and transition the destination address back to active state. However, there may be a situation where HEARTBEAT chunks can go through while DATA chunks cannot. Hence, in a situation where a HEARTBEAT ACK arrives while there is data outstanding towards the destination address to which the HEARTBEAT was sent, then an implementation

MAY choose to not have the HEARTBEAT ACK reset the error counter, but have the error counter reset await the fate of the outstanding data transmission. This situation can happen when data is sent to a destination address in PF state. When the sender resumes data transmission on a destination address after a transition of the destination address from PF to active state, it MUST do this following the prescriptions of Section 7.2 of [RFC4960].

8. Additional (PMR - PFMR) consecutive timeouts on a destination address in PF state confirm the path failure, upon which the destination address transitions to the inactive state. As described in [RFC4960], the sender (i) SHOULD notify the ULP about this state transition, and (ii) transmit HEARTBEAT chunks to the inactive destination address at a lower HB.interval frequency as described in Section 8.3 of [RFC4960] (when the Path Heartbeat function is enabled for the destination address).
9. Acknowledgments for chunks that have been transmitted to multiple destinations (i.e., a chunk which has been retransmitted to a different destination address than the destination address to which the chunk was first transmitted) SHOULD NOT clear the error count for an inactive destination address and SHOULD NOT move a destination address in PF state back to active state, since a sender cannot disambiguate whether the ACK was for the original transmission or the retransmission(s). A SCTP sender MAY clear the error counter and move a destination address back to active state by information other than acknowledgments, when it can uniquely determine which destination, among multiple destination addresses, the chunk reached. This document makes no reference to what such information could consist of, nor how such information could be obtained.
10. Acknowledgments for data chunks that has been transmitted to one destination address only MUST clear the error counter for the destination address and MUST transition a destination address in PF state back to active state. This situation can happen when new data is sent to a destination address in the PF state. It can also happen in situations where the destination address is in the PF state due to the occurrence of a spurious T3-rtx timer and acknowledgments start to arrive for data sent prior to occurrence of the spurious T3-rtx and data has not yet been retransmitted towards other destinations. This document does not specify special handling for detection of or reaction to spurious T3-rtx timeouts, e.g., for special operation vis-a-vis the congestion control handling or data retransmission operation towards a destination address which undergoes a transition from

active to PF to active state due to a spurious T3-rtx timeout. But it is noted that this is an area which would benefit from additional attention, experimentation and specification for single-homed SCTP as well as for multi-homed SCTP protocol operation.

11. When all destination addresses are in inactive state, and SCTP protocol operation thus is said to be in dormant state, the prescriptions given in Section 4 shall be followed.
12. The SCTP stack SHOULD expose the PF state of its destination addresses to the ULP as well as provide the means to notify the ULP of state transitions of its destination addresses from active to PF, and vice-versa. However it is recommended that an SCTP stack implementing SCTP-PF also allows for that the ULP is kept ignorant of the PF state of its destinations and the associated state transitions, thus allowing for retain of the simpler state transition model of RFC4960 in the ULP. For this reason it is recommended that an SCTP stack implementing SCTP-PF also provides the ULP with the means to suppress exposure of the PF state and the associated state transitions.

4. Dormant State Operation

In a situation with complete disruption of the communication in between the SCTP Endpoints, the aggressive HEARTBEAT transmissions of SCTP-PF on destination addresses in PF state may make the association enter dormant state faster than a standard [RFC4960] SCTP implementation given the same setting of Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR). For example, an SCTP association with two destination addresses typically would reach dormant state in half the time of an [RFC4960] SCTP implementation in such situations. This is because a SCTP PF sender will send HEARTBEATS and data retransmissions in parallel with RTO intervals when there are multiple destinations addresses in PF state. This argument presumes that $RTO \ll HB.interval$ of [RFC4960]. With the design goal that SCTP-PF shall provide the same level of disruption tolerance as an [RFC4960] SCTP implementation with the same Path.Max.Retrans (PMR) and Association.Max.Retrans (AMR) setting, we prescribe for that an SCTP-PF implementation SHOULD operate as described below in Section 4.1 during dormant state.

An SCTP-PF implementation MAY choose a different dormant state operation than the one described below in Section 4.1 provided that the solution chosen does not decrease the fault tolerance of the SCTP-PF operation.

The below prescription for SCTP-PF dormant state handling MUST NOT be coupled to the value of the PFMR, but solely to the activation of SCTP-PF logic in an SCTP implementation.

It is noted that the below dormant state operation is considered to provide added disruption tolerance also for an [RFC4960] SCTP implementation, and that it can be sensible for an [RFC4960] SCTP implementation to follow this mode of operation. For an [RFC4960] SCTP implementation the continuation of data transmission during dormant state makes the fault tolerance of SCTP be more robust towards situations where some, or all, alternative paths of an SCTP association approach, or reach, inactive state before the primary path used for data transmission observes trouble.

4.1. SCTP Dormant State Procedure

- a. When the destination addresses are all in inactive state and data is available for transfer, the sender MUST choose one destination and transmit data to this destination address.
- b. The sender MUST NOT change the state of the chosen destination address (it remains in inactive state) and it MUST NOT clear the error counter of the destination address as a result of choosing the destination address for data transmission.
- c. The sender SHOULD choose the destination in inactive state with the lowest error count (fewest consecutive timeouts) for data transmission. When there are multiple destinations with same error count in inactive state, the sender SHOULD attempt to pick the most divergent source - destination pair from the last source - destination pair where failure was observed. Rules for picking the most divergent source-destination pair are an implementation decision and are not specified within this document. To support differentiation of inactive destination addresses based on their error count SCTP will need to allow for increment of the destination address error counters up to some reasonable limit above PMR+1, thus changing the prescriptions of [RFC4960], section 8.3, in this respect. The exact limit to apply is not specified in this document but it is considered reasonable to require for the limit to be an order of magnitude higher than the PMR value. A sender MAY choose to deploy other strategies than the strategy defined here. The strategy to prioritize the last active destination address, i.e., the destination address with the fewest error counts is optimal when some paths are permanently inactive, but suboptimal when a path instability is transient.

5. Primary Path Switchover

The objective of the Primary Path Switchover operation is to allow the SCTP sender to continue data transmission on a new working path even when the old primary destination address becomes active again. This is achieved by having SCTP perform a switchover of the primary path to the new working path if the error counter of the primary path exceeds a certain threshold. This mode of operation can be applied not only to SCTP-PF implementations, but also to [RFC4960] implementations.

The Primary Path Switchover operation requires only sender side changes. The details are:

1. The sender maintains a new tunable parameter, called Primary.Switchover.Max.Retrans (PSMR). For SCTP-PF implementations, the PSMR MUST be set greater or equal to the PFMR value. For [RFC4960] implementations the PSMR MUST be set greater or equal to the PMR value. Implementations MUST reject any other values of PSMR.
2. When the path error counter on a set primary path exceeds PSMR, the SCTP implementation MUST autonomously select and set a new primary path.
3. The primary path selected by the SCTP implementation MUST be the path which at the given time would be chosen for data transfer. A previously failed primary path can be used as data transfer path as per normal path selection when the present data transfer path fails.
4. For SCTP-PF, the recommended value of PSMR is PFMR when Primary Path Switchover operation mode is used. This means that no forced switchback to a previously failed primary path is performed. An SCTP-PF implementation of Primary Path Switchover MUST support the setting of PSMR = PFMR. A SCTP-PF implementation of Primary Path Switchover MAY support setting of PSMR > PFMR.
5. For [RFC4960] SCTP, the recommended value of PSMR is PMR when Primary Path Switchover is used. This means that no forced switchback to a previously failed primary path is performed. A [RFC4960] SCTP implementation of Primary Path Switchover MUST support the setting of PSMR = PMR. An [RFC4960] SCTP implementation of Primary Path Switchover MAY support larger settings of PSMR > PMR.

6. It MUST be possible to disable the Primary Path Switchover operation and obtain the standard switchback operation of [RFC4960].

The manner of switchover operation that is most optimal in a given scenario depends on the relative quality of a set primary path versus the quality of alternative paths available as well as on the extent to which it is desired for the mode of operation to enforce traffic distribution over a number of network paths. I.e., load distribution of traffic from multiple SCTP associations may be sought to be enforced by distribution of the set primary paths with [RFC4960] switchback operation. However as [RFC4960] switchback behavior is suboptimal in certain situations, especially in scenarios where a number of equally good paths are available, an SCTP implementation MAY support also, as alternative behavior, the Primary Path Switchover mode of operation and MAY enable it based on applications' requests.

For an SCTP implementation that implements the Primary Path Switchover operation, this specification RECOMMENDS that the standard RFC4960 switchback operation is retained as the default operation.

6. Suggested SCTP Protocol Parameter Values

This document does not alter the [RFC4960] value recommendation for the SCTP Protocol Parameters defined in [RFC4960].

The following protocol parameter is RECOMMENDED:

PotentiallyFailed.Max.Retrans (PFMR) - 0

7. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to control and observe the SCTP-PF behavior as well as the Primary Path Switchover function.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is, by means of the existing SCTP_PEER_ADDR_CHANGE event, extended to provide the event notification when a peer address enters or leaves the potentially failed state as well as the socket API implementation is extended to expose the potentially failed state of a peer address in the existing SCTP_GET_PEER_ADDR_INFO structure.

Furthermore, two new read/write socket options for the level IPPROTO_SCTP and the name SCTP_PEER_ADDR_THLDS and

SCTP_EXPOSE_POTENTIALLY_FAILED_STATE are defined as described below. The first socket option is used to control the values of the PFMR and PSMP parameters described in Section 3 and in Section 5. The second one controls the exposition of the potentially failed path state.

Support for the SCTP_PEER_ADDR_THLDS and SCTP_EXPOSE_POTENTIALLY_FAILED_STATE socket options need also to be added to the function sctp_opt_info().

7.1. Support for the Potentially Failed Path State

As defined in [RFC6458], the SCTP_PEER_ADDR_CHANGE event is provided if the status of a peer address changes. In addition to the state changes described in [RFC6458], this event is also provided, if a peer address enters or leaves the potentially failed state. The notification as defined in [RFC6458] uses the following structure:

```
struct sctp_paddr_change {
    uint16_t spc_type;
    uint16_t spc_flags;
    uint32_t spc_length;
    struct sockaddr_storage spc_aaddr;
    uint32_t spc_state;
    uint32_t spc_error;
    sctp_assoc_t spc_assoc_id;
}
```

[RFC6458] defines the constants SCTP_ADDR_AVAILABLE, SCTP_ADDR_UNREACHABLE, SCTP_ADDR_REMOVED, SCTP_ADDR_ADDED, and SCTP_ADDR_MADE_PRIM to be provided in the spc_state field. This document defines in addition to that the new constant SCTP_ADDR_POTENTIALLY_FAILED, which is reported if the affected address becomes potentially failed.

The SCTP_GET_PEER_ADDR_INFO socket option defined in [RFC6458] can be used to query the state of a peer address. It uses the following structure:

```
struct sctp_paddrinfo {
    sctp_assoc_t spinfo_assoc_id;
    struct sockaddr_storage spinfo_address;
    int32_t spinfo_state;
    uint32_t spinfo_cwnd;
    uint32_t spinfo_srtt;
    uint32_t spinfo_rto;
    uint32_t spinfo_mtu;
};
```


[RFC6458] defines the constants `SCTP_UNCONFIRMED`, `SCTP_ACTIVE`, and `SCTP_INACTIVE` to be provided in the `spinfo_state` field. This document defines in addition to that the new constant `SCTP_POTENTIALLY_FAILED`, which is reported if the peer address is potentially failed.

7.2. Peer Address Thresholds (`SCTP_PEER_ADDR_THLDS`) Socket Option

Applications can control the SCTP-PF behavior by getting or setting the number of consecutive timeouts before a peer address is considered potentially failed or unreachable. The same socket option is used by applications to set and get the number of timeouts before the primary path is changed automatically by the Primary Path Switchover function. This socket option uses the level `IPPROTO_SCTP` and the name `SCTP_PEER_ADDR_THLDS`.

The following structure is used to access and modify the thresholds:

```
struct sctp_paddrthlds {
    sctp_assoc_t spt_assoc_id;
    struct sockaddr_storage spt_address;
    uint16_t spt_pathmaxrxt;
    uint16_t spt_pathpfthld;
    uint16_t spt_pathcpthld;
};
```

`spt_assoc_id`: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application may fill in an association identifier or `SCTP_FUTURE_ASSOC`. It is an error to use `SCTP_{CURRENT|ALL}_ASSOC` in `spt_assoc_id`.

`spt_address`: This specifies which peer address is of interest. If a wild card address is provided, this socket option applies to all current and future peer addresses.

`spt_pathmaxrxt`: Each peer address of interest is considered unreachable, if its path error counter exceeds `spt_pathmaxrxt`.

`spt_pathpfthld`: Each peer address of interest is considered Potentially Failed, if its path error counter exceeds `spt_pathpfthld`.

`spt_pathcpthld`: Each peer address of interest is not considered the primary remote address anymore, if its path error counter exceeds `spt_pathcpthld`. Using a value of `0xffff` disables the selection of a new primary peer address. If an implementation does not support the automatically selection of a new primary address, it should indicate an error with `errno` set to `EINVAL` if a value different

from 0xffff is used in `spt_pathcpthld`. For SCTP-PF, the setting of `spt_pathcpthld < spt_pathpfthld` should be rejected with `errno` set to `EINVAL`. For [RFC4960] SCTP, the setting of `spt_pathcpthld < spt_pathmaxrxt` should be rejected with `errno` set to `EINVAL`. A SCTP-PF implementation may support only setting of `spt_pathcpthld = spt_pathpfthld` and `spt_pathcpthld = 0xffff` and a [RFC4960] SCTP implementation may support only setting of `spt_pathcpthld = spt_pathmaxrxt` and `spt_pathcpthld = 0xffff`. In these cases SCTP shall reject setting of other values with `errno` set to `EINVAL`.

7.3. Exposing the Potentially Failed Path State (`SCTP_EXPOSE_POTENTIALLY_FAILED_STATE`) Socket Option

Applications can control the exposure of the potentially failed path state in the `SCTP_PEER_ADDR_CHANGE` event and the `SCTP_GET_PEER_ADDR_INFO` as described in Section 7.1. The default value is implementation specific.

This socket option uses the level `IPPROTO_SCTP` and the name `SCTP_EXPOSE_POTENTIALLY_FAILED_STATE`.

The following structure is used to control the exposition of the potentially failed path state:

```
struct sctp_assoc_value {
    sctp_assoc_t assoc_id;
    uint32_t assoc_value;
};
```

`assoc_id`: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application may fill in an association identifier or `SCTP_FUTURE_ASSOC`. It is an error to use `SCTP_{CURRENT|ALL}_ASSOC` in `assoc_id`.

`assoc_value`: The potentially failed path state is exposed if and only if this parameter is non-zero.

8. Security Considerations

Security considerations for the use of SCTP and its APIs are discussed in [RFC4960] and [RFC6458].

The logic introduced by this document does not impact existing SCTP messages on the wire. Also, this document does not introduce any new SCTP messages on the wire that require new security considerations.

SCTP-PF makes SCTP not only more robust during primary path failure/congestion but also more vulnerable to network connectivity/

congestion attacks on the primary path. SCTP-PF makes it easier for an attacker to trick SCTP to change data transfer path, since the duration of time that an attacker needs to negatively influence the network connectivity is much shorter than [RFC4960]. However, SCTP-PF does not constitute a significant change in the duration of time and effort an attacker needs to keep SCTP away from the primary path. With the standard switchback operation [RFC4960] SCTP resumes data transfer on its primary path as soon as the next HEARTBEAT succeeds.

On the other hand, usage of the Primary Path Switchover mechanism, does change the threat analysis. This is because on-path attackers can force a permanent change of the data transfer path by blocking the primary path until the switchover of the primary path is triggered by the Primary Path Switchover algorithm. This especially will be the case when the Primary Path Switchover is used together with SCTP-PF with the particular setting of PSMR = PFMR = 0, as Primary Path Switchover here happens already at the first RTO timeout experienced. Users of the Primary Path Switchover mechanism should be aware of this fact.

The event notification of path state transfer from active to potentially failed state and vice versa gives attackers an increased possibility to generate more local events. However, it is assumed that event notifications are rate-limited in the implementation to address this threat.

9. MIB Considerations

SCTP-PF introduces new SCTP algorithms for failover and switchback with associated new state parameters. It is recommended that the SCTP-MIB defined in [RFC3873] is updated to support the management of the SCTP-PF implementation. This can be done by extending the sctpAssocRemAddrActive field of the SCTPAssocRemAddrTable to include information of the PF state of the destination address and by adding new fields to the SCTPAssocRemAddrTable supporting PotentiallyFailed.Max.Retrans (PFMR) and Primary.Switchover.Max.Retrans (PSMR) parameters.

10. IANA Considerations

This document does not create any new registries or modify the rules for any existing registries managed by IANA.

11. Acknowledgements

The authors wish to thank Michael Tuexen for his many invaluable comments and for his very substantial support with the making of this document.

12. Proposed Change of Status (to be Deleted before Publication)

Initially this work looked to entail some changes of the Congestion Control (CC) operation of SCTP and for this reason the work was proposed as Experimental. These intended changes of the CC operation have since been judged to be irrelevant and are no longer part of the specification. As the specification entails no other potential harmful features, consensus exists in the WG to bring the work forward as PS.

Initially concerns have been expressed about the possibility for the mechanism to introduce path bouncing with potential harmful network impacts. These concerns are believed to be unfounded. This issue is addressed in Appendix B.

It is noted that the feature specified by this document is implemented by multiple SCTP SW implementations and furthermore that various variants of the solution have been deployed in telephony signaling environments for several years with good results.

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

13.2. Informative References

- [CARO02] Caro Jr., A., Iyengar, J., Amer, P., Heinz, G., and R. Stewart, "A Two-level Threshold Recovery Mechanism for SCTP", Tech report, CIS Dept, University of Delaware , 7 2002.
- [CARO04] Caro Jr., A., Amer, P., and R. Stewart, "End-to-End Failover Thresholds for Transport Layer Multi homing", MILCOM 2004 , 11 2004.
- [CARO05] Caro Jr., A., "End-to-End Fault Tolerance using Transport Layer Multi homing", Ph.D Thesis, University of Delaware , 1 2005.

[FALLON08]

Fallon, S., Jacob, P., Qiao, Y., Murphy, L., Fallon, E., and A. Hanley, "SCTP Switchover Performance Issues in WLAN Environments", IEEE CCNC 2008, 1 2008.

[GRINNEMO04]

Grinnemo, K-J. and A. Brunstrom, "Performance of SCTP-controlled failovers in M3UA-based SIGTRAN networks", Advanced Simulation Technologies Conference , 4 2004.

[IYENGAR06]

Iyengar, J., Amer, P., and R. Stewart, "Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-end Paths.", IEEE/ACM Trans on Networking 14(5), 10 2006.

[JUNGMAIER02]

Jungmaier, A., Rathgeb, E., and M. Tuexen, "On the use of SCTP in failover scenarios", World Multiconference on Systemics, Cybernetics and Informatics , 7 2002.

[NATARAJAN09]

Natarajan, P., Ekiz, N., Amer, P., and R. Stewart, "Concurrent Multipath Transfer during Path Failure", Computer Communications , 5 2009.

[RFC3873]

Pastor, J. and M. Belinchon, "Stream Control Transmission Protocol (SCTP) Management Information Base (MIB)", RFC 3873, DOI 10.17487/RFC3873, September 2004, <<http://www.rfc-editor.org/info/rfc3873>>.

[RFC6458]

Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, December 2011.

Appendix A. Discussions of Alternative Approaches

This section lists alternative approaches for the issues described in this document. Although these approaches do not require to update RFC4960, we do not recommend them from the reasons described below.

A.1. Reduce Path.Max.Retrans (PMR)

Smaller values for Path.Max.Retrans shorten the failover duration and in fact this is recommended in some research results [JUNGMAIER02] [GRINNEMO04] [FALLON08]. However to significantly reduce the failover time it is required to go down (as with PFMR) to Path.Max.Retrans=0 and with this setting SCTP switches to another

destination address already on a single timeout which may result in spurious failover. Spurious failover is a problem in [RFC4960] SCTP as the transmission of HEARTBEATS on the left primary path, unlike in SCTP-PF, is governed by 'HB.interval' also during the failover process. 'HB.interval' is usually set in the order of seconds (recommended value is 30 seconds) and when the primary path becomes inactive, the next HEARTBEAT may be transmitted only many seconds later. Indeed as recommended, only 30 secs later. Meanwhile, the primary path may since long have recovered, if it needed recovery at all (indeed the failover could be truly spurious). In such situations, post failover, an endpoint is forced to wait in the order of many seconds before the endpoint can resume transmission on the primary path and furthermore once it returns on the primary path the CWND needs to be rebuild anew - a process which the throughput already have had to suffer from on the alternate path. Using a smaller value for 'HB.interval' might help this situation, but it would result in a general waste of bandwidth as such more frequent HEARTBEATING would take place also when there are no observed troubles. The bandwidth overhead may be diminished by having the ULP use a smaller 'HB.interval' only on the path which at any given time is set to be the primary path, but this adds complication in the ULP.

In addition, smaller Path.Max.Retrans values also affect the 'Association.Max.Retrans' value. When the SCTP association's error count exceeds Association.Max.Retrans threshold, the SCTP sender considers the peer endpoint unreachable and terminates the association. Section 8.2 in [RFC4960] recommends that Association.Max.Retrans value should not be larger than the summation of the Path.Max.Retrans of each of the destination addresses. Else the SCTP sender considers its peer reachable even when all destinations are INACTIVE and to avoid this dormant state operation, [RFC4960] SCTP implementation SHOULD reduce Association.Max.Retrans accordingly whenever it reduces Path.Max.Retrans. However, smaller Association.Max.Retrans value decreases the fault tolerance of SCTP as it increases the chances of association termination during minor congestion events.

A.2. Adjust RTO related parameters

As several research results indicate, we can also shorten the duration of failover process by adjusting RTO related parameters [JUNGMAIER02] [FALLON08]. During failover process, RTO keeps being doubled. However, if we can choose smaller value for RTO.max, we can stop the exponential growth of RTO at some point. Also, choosing smaller values for RTO.initial or RTO.min can contribute to keep the RTO value small.

Similar to reducing Path.Max.Retrans, the advantage of this approach is that it requires no modification to the current specification, although it needs to ignore several recommendations described in the Section 15 of [RFC4960]. However, this approach requires to have enough knowledge about the network characteristics between end points. Otherwise, it can introduce adverse side-effects such as spurious timeouts.

The significant issue with this approach, however, is that even if the RTO.max is lowered to an optimal low value, then as long as the Path.Max.Retrans is kept at the [RFC4960] recommended value, the reduction of the RTO.max doesn't reduce the failover time sufficiently enough to prevent severe performance degradation during failover.

Appendix B. Discussions for Path Bouncing Effect

The methods described in the document can accelerate the failover process. Hence, they might introduce the path bouncing effect where the sender keeps changing the data transmission path frequently. This sounds harmful to the data transfer, however several research results indicate that there is no serious problem with SCTP in terms of path bouncing effect [CARO04] [CARO05].

There are two main reasons for this. First, SCTP is basically designed for multipath communication, which means SCTP maintains all path related parameters (CWND, ssthresh, RTT, error count, etc) per each destination address. These parameters cannot be affected by path bouncing. In addition, when SCTP migrates the data transfer to another path, it starts with the minimal or the initial CWND. Hence, there is little chance for packet reordering or duplicating.

Second, even if all communication paths between the end-nodes share the same bottleneck, the SCTP-PF results in a behavior already allowed by [RFC4960].

Appendix C. SCTP-PF for SCTP Single-homed Operation

For a single-homed SCTP association the only tangible effect of the activation of SCTP-PF operation is enhanced failure detection in terms of potential notification of the PF state of the sole destination address as well as, for idle associations, more rapid entering, and notification, of inactive state of the destination address and more rapid end-point failure detection. It is believed that neither of these effects are harmful, provided adequate dormant state operation is implemented, and furthermore that they may be particularly useful for applications that deploys multiple SCTP associations for load balancing purposes. The early notification of

the PF state may be used for preventive measures as the entering of the PF state can be used as a warning of potential congestion. Depending on the PMR value, the aggressive HEARTBEAT transmission in PF state may speed up the end-point failure detection (exceed of AMR threshold on the sole path error counter) on idle associations in case where relatively large HB.interval value compared to RTO (e.g. 30secs) is used.

Authors' Addresses

Yoshifumi Nishida
GE Global Research
2623 Camino Ramon
San Ramon, CA 94583
USA

Email: nishida@wide.ad.jp

Preethi Natarajan
Cisco Systems
510 McCarthy Blvd
Milpitas, CA 95035
USA

Email: prenatar@cisco.com

Armando Caro
BBN Technologies
10 Moulton St.
Cambridge, MA 02138
USA

Email: acar@bbn.com

Paul D. Amer
University of Delaware
Computer Science Department - 434 Smith Hall
Newark, DE 19716-2586
USA

Email: amer@udel.edu

Karen E. E. Nielsen
Ericsson
Kistavaegen 25
Stockholm 164 80
Sweden

Email: karen.nielsen@tieto.com

Network Working Group	M. Tuexen
Internet-Draft	I. Ruengeler
Updates: 4960 (if approved)	Muenster Univ. of Appl. Sciences
Intended status: Standards Track	R. Stewart
Expires: March 01, 2014	Adara Networks
	August 28, 2013

SACK-IMMEDIATELY Extension for the Stream Control Transmission Protocol
draft-ietf-tsvwg-sctp-sack-immediately-04.txt

Abstract

This document updates RFC 4960 by defining a method for the sender of a DATA chunk to indicate that the corresponding SACK chunk should be sent back immediately and not be delayed. It is done by specifying a bit in the DATA chunk header, called the I-bit, which can get set either by the SCTP implementation or by the application using an SCTP stack. Since unknown flags in chunk headers are ignored by SCTP implementations, this extension does not introduce any interoperability problems.

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 01, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions	3
3. The I-bit in the DATA Chunk Header	3
4. Use Cases	4
4.1. Triggering at the Application Level	4
4.2. Triggering at the SCTP Level	4
5. Procedures	4
5.1. Sender Side Considerations	5
5.2. Receiver Side Considerations	5
6. Interoperability Considerations	5
7. Socket API Considerations	5
8. IANA Considerations	6
9. Security Considerations	7
10. Acknowledgments	7
11. References	7
11.1. Normative References	7
11.2. Informative References	7
Authors' Addresses	7

1. Introduction

According to [RFC4960] the receiver of a DATA chunk should use delayed SACKs. This delaying is completely controlled by the receiver of the DATA chunk and remains the default behavior.

In specific situations the delaying of SACKs results in reduced performance of the protocol:

1. If such a situation can be detected by the receiver, the corresponding SACK can be sent immediately. For example, [RFC4960] recommends the immediate sending if the receiver has detected message loss or message duplication.

2. However, if the situation can only be detected by the sender of the DATA chunk, [RFC4960] provides no method of avoiding a delay in sending the SACK. Examples of these situations include ones which require interaction with the application (e.g. applications using the SCTP_SENDER_DRY_EVENT, see Section 4.1) and ones which can be detected by the SCTP stack itself (e.g. closing the association, hitting window limits or resetting streams, see Section 4.2).

To overcome the limitation described in the second case, this document describes a simple extension of the SCTP DATA chunk by defining a new flag, the I-bit. The sender of a DATA chunk indicates by setting this bit that the corresponding SACK chunk should not be delayed.

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. The I-bit in the DATA Chunk Header

The following Figure 1 shows the extended DATA chunk.

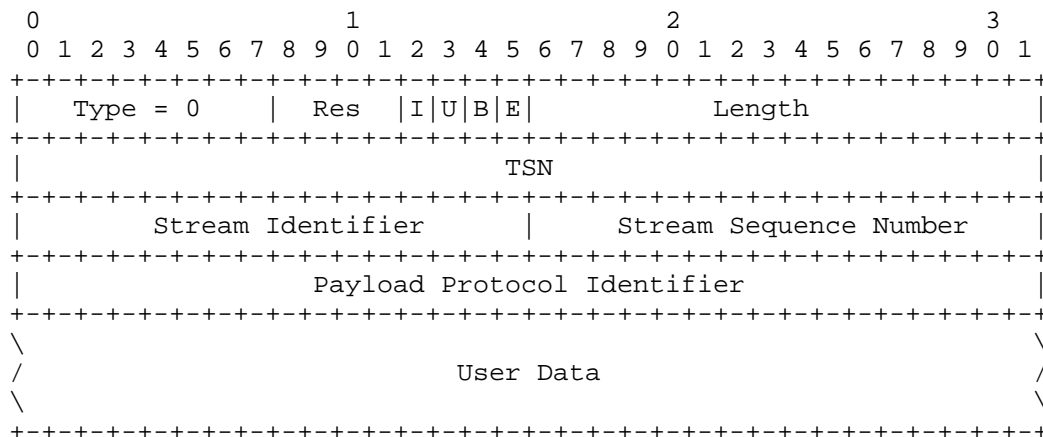


Figure 1: Extended DATA chunk format

The only difference between the DATA chunk in Figure 1 and the DATA chunk defined in [RFC4960] is the addition of the I-bit in the flags field of the DATA chunk header.

This bit was Reserved in [RFC4960]. [RFC4960] specified that this bit should be set to 0 by the sender and ignored by the receiver.

4. Use Cases

The setting of the I-bit can either be triggered by the application using SCTP or by the SCTP stack itself. The following two subsections provide a non-exhaustive list of examples.

4.1. Triggering at the Application Level

One example of a situation in which it may be desirable for an application to trigger setting of the I-bit involves the SCTP_SENDER_DRY_EVENT in the SCTP socket API [RFC6458]. Upper layers of SCTP using the socket API as defined in [RFC6458] may subscribe to the SCTP_SENDER_DRY_EVENT for getting a notification as soon as no user data is outstanding anymore. To avoid an unnecessary delay while waiting for such an event, the application can request the setting of the I-Bit when sending the last user message before waiting for the event. This results in setting the I-bit of the last DATA chunk corresponding to the user message and is possible using the extension of the socket API described in Section 7.

4.2. Triggering at the SCTP Level

There are also situations in which the SCTP implementation can set the I-bit without interacting with the upper layer.

If the association is in the SHUTDOWN-PENDING state, setting the I-bit reduces the number of simultaneous associations for a busy server handling short living associations.

Another case is where the sending of a DATA chunk fills the congestion or receiver window. Setting the I-bit in these cases improves the throughput of the transfer.

If an SCTP association supports the SCTP Stream Reconfiguration extension defined in [RFC6525], the performance can be improved by setting the I-bit when there are pending reconfiguration requests that require that there be no outstanding DATA chunks.

5. Procedures

5.1. Sender Side Considerations

Whenever the sender of a DATA chunk can benefit from the corresponding SACK chunk being sent back without delay, the sender MAY set the I-bit in the DATA chunk header. Please note that it is irrelevant to the receiver why the sender has set the I-bit.

Reasons for setting the I-bit include, but are not limited to, the following (see Section 4 for the benefits):

- o The application requests to set the I-bit of the last DATA chunk of a user message when providing the user message to the SCTP implementation (see Section 7).
- o The sender is in the SHUTDOWN-PENDING state.
- o The sending of a DATA chunk fills the congestion or receiver window.
- o The sending of an Outgoing SSN Reset Request Parameter or an SSN/TSN Reset Request Parameter is pending, if the association supports the Stream Reconfiguration extension defined in [RFC6525].

5.2. Receiver Side Considerations

On reception of an SCTP packet containing a DATA chunk with the I-bit set, the receiver SHOULD NOT delay the sending of the corresponding SACK chunk, i.e., the receiver SHOULD immediately respond with the corresponding SACK chunk.

6. Interoperability Considerations

According to [RFC4960] the receiver of a DATA chunk with the I-bit set should ignore this bit when it does not support the extension described in this document. Since the sender of the DATA chunk is able to handle this case, there is no requirement for negotiating the support of the feature described in this document.

7. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to set the I-bit.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] needs to be extended to allow the application to set the I-bit of the last DATA chunk when sending each user message.

This can be done by setting a flag called `SCTP_SACK_IMMEDIATELY` in the `snd_flags` field of the struct `sctp_sndinfo` structure when using `sctp_sendv()` or `sendmsg()`. If the deprecated struct `sctp_sndrcvinfo` structure is used instead when calling `sctp_send()`, `sctp_sendx()`, or `sendmsg()`, the `SCTP_SACK_IMMEDIATELY` flag can be set in the `sinfo_flags` field. When using the deprecated function `sctp_sendmsg()` the `SCTP_SACK_IMMEDIATELY` flag can be in the `flags` parameter.

8. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

Following the chunk flag registration procedure defined in [RFC6096], IANA should register a new bit, the I-bit, for the DATA chunk. The suggested value is 0x08 and the reference should be RFCXXXX.

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

DATA Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	E bit	[RFC4960]
0x02	B bit	[RFC4960]
0x04	U bit	[RFC4960]
0x08	I Bit	[RFCXXXX]
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

9. Security Considerations

See [RFC4960] for general security considerations for SCTP. In addition, a malicious sender can force its peer to send packets containing a SACK chunk for each received packet containing DATA chunks instead of every other. This could impact the network, resulting in more packets sent on the network, or the peer because the generating and sending of the packets has some processing cost. However, the additional packets can only contain the most simplest SACK chunk (no gap reports, no duplicate TSNs), since in case of packet drop or reordering in the network a SACK chunk would be sent immediately anyway. Therefore this does neither introduce a significant additional processing cost on the receiver side. This does not result in more traffic in the network than a receiver that sends a SACK for every packet, which is already permitted.

10. Acknowledgments

The authors wish to thank Mark Allmann, Brian Bidulock, David Black, Anna Brunstrom, Gorry Fairhurst, Janardhan Iyengar, Kacheong Poon, and Michael Welzl for their invaluable comments.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, January 2011.

11.2. Informative References

- [RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, December 2011.
- [RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, February 2012.

Authors' Addresses

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstr. 39
48565 Steinfurt
DE

Email: tuexen@fh-muenster.de

Irene Ruengeler
Muenster University of Applied Sciences
Stegerwaldstr. 39
48565 Steinfurt
DE

Email: i.ruengeler@fh-muenster.de

Randall R. Stewart
Adara Networks
Chapin, SC 29036
US

Email: randall@lakerest.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 20, 2013

M. Tuexen
Muenster Univ. of Appl. Sciences
R. R. Stewart
Adara Networks
March 19, 2013

UDP Encapsulation of SCTP Packets for End-Host to End-Host Communication
draft-ietf-tsvwg-sctp-udp-encaps-14.txt

Abstract

This document describes a simple method of encapsulating SCTP Packets into UDP packets and its limitations. This allows the usage of SCTP in networks with legacy NAT not supporting SCTP. It can also be used to implement SCTP on hosts without directly accessing the IP-layer, for example implementing it as part of the application without requiring special privileges.

Please note that this document only describes the functionality required within an SCTP stack to add on UDP encapsulation, providing only those mechanisms for two end-hosts to communicate with each other over UDP ports. In particular, it does not provide mechanisms to determine whether UDP encapsulation is being used by the peer, nor the mechanisms for determining which remote UDP port number can be used. These functions are out of scope for this document.

This document covers only end-hosts and not tunneling (egress or ingress) end-points.

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 20, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Conventions	3
3. Use Cases	3
3.1. Portable SCTP Implementations	3
3.2. Legacy NAT Traversal	4
4. Unilateral Self-Address Fixing (UNSAF) Considerations	4
5. SCTP over UDP	4
5.1. Architectural Considerations	4
5.2. Packet Format	5
5.3. Encapsulation Procedure	6
5.4. Decapsulation Procedure	7
5.5. ICMP Considerations	7
5.6. Path MTU Considerations	8
5.7. Handling of Embedded IP-addresses	8
5.8. ECN Considerations	8
6. Socket API Considerations	8
6.1. Get or Set the Remote UDP Encapsulation Port Number (SCTP_REMOTE_UDP_ENCAPS_PORT)	9
7. IANA Considerations	9
8. Security Considerations	9
9. Acknowledgments	10
10. References	10
10.1. Normative References	10
10.2. Informative References	11
Authors' Addresses	11

1. Introduction

This document describes a simple method of encapsulating SCTP packets into UDP packets. SCTP as defined in [RFC4960] runs directly over IPv4 or IPv6. There are two main reasons for encapsulating SCTP packets:

- o Allow SCTP traffic to pass through legacy NATs, which do not provide native SCTP support as specified in [I-D.ietf-behave-sctpnat] and [I-D.ietf-tsvwg-natsupp].
- o Allow SCTP to be implemented on hosts which do not provide direct access to the IP-layer. In particular, applications can use their own SCTP implementation if the operating system does not provide one.

SCTP provides the necessary congestion control and reliability service that UDP does not perform.

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. Use Cases

This section discusses two important use cases for encapsulating SCTP into UDP.

3.1. Portable SCTP Implementations

Some operating systems support SCTP natively. For other operating systems implementations are available, but require special privileges to install and/or use them. In some cases no kernel implementation might be available at all. When providing an SCTP implementation as part of a user process, most operating systems require special privileges to access the IP layer directly.

Using UDP encapsulation makes it possible to provide an SCTP implementation as part of a user process which does not require any special privileges.

A crucial point for implementing SCTP in user space is that the source address of outgoing packets needs to be controlled. This is not an issue if the SCTP stack can use all addresses configured at the IP-layer as source addresses. However, it is an issue when also using the address management required for NAT traversal, described in Section 5.7.

3.2. Legacy NAT Traversal

Using UDP encapsulation allows SCTP communication when traversing legacy NATs (i.e those NATs not supporting SCTP as described in [I-D.ietf-behave-sctpnat] and [I-D.ietf-tsvwg-natsupp]). For single-homed associations IP addresses MUST NOT be listed in the INIT and INIT-ACK chunks. To use multiple addresses, the dynamic address reconfiguration extension described in [RFC5061] MUST be used only with wildcard addresses in the ASCONF chunks in combination with [RFC4895].

For multi-homed SCTP association the address management as described in Section 5.7 MUST be performed.

SCTP sends periodic HEARTBEAT chunks on all idle paths. These can keep the NAT state alive.

4. Unilateral Self-Address Fixing (UNSAF) Considerations

As [RFC3424] requires a limited scope, this document only covers SCTP end-points dealing with legacy constraints as described in Section 3. It doesn't cover generic tunneling end-points.

Obviously, the exit strategy is to use hosts supporting SCTP natively and middleboxes supporting SCTP as specified in [I-D.ietf-behave-sctpnat] and [I-D.ietf-tsvwg-natsupp]).

5. SCTP over UDP

5.1. Architectural Considerations

UDP encapsulated SCTP is normally communicated between SCTP stacks using the IANA-assigned UDP port number 9899 (sctp-tunneling) on both ends. There are circumstances where other ports may be used on either end: As stated earlier, implementations in the application space might be required to use other than the registered port. Since NAT boxes might change UDP port numbers, the receiver might observe other UDP port numbers than were used by the sender. Discovery of alternate ports is outside of the scope of this document, but this section describes considerations for SCTP stack design in light of their potential use.

Each SCTP stack uses a single local UDP encapsulation port number as the destination port for all its incoming SCTP packets. While the uniqueness of the local UDP encapsulation port number is not necessarily required for the protocol, this greatly simplifies implementation design, since different ports for each address would require a sender implementation to choose the appropriate port while doing source address selection. Using a single local UDP encapsulation port number per host is not possible if the SCTP stack is implemented as part of each application, there are multiple applications, and some of the applications want to use the same IP-address.

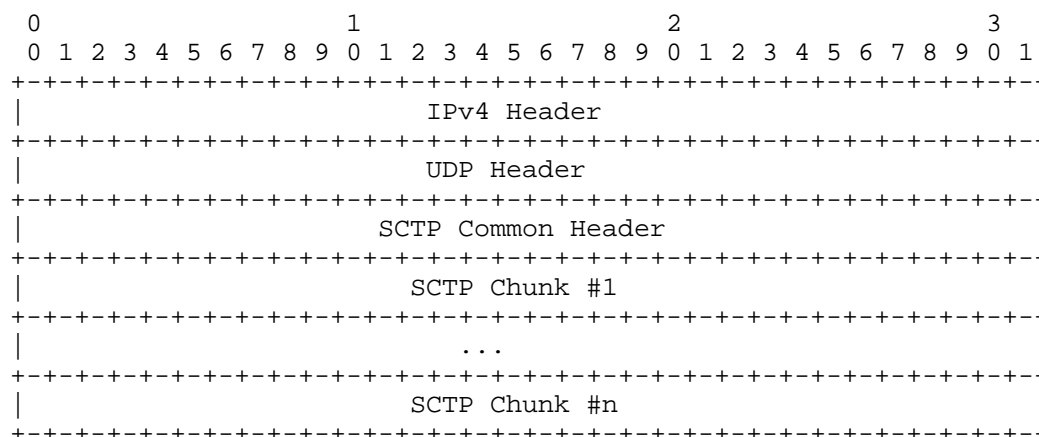
An SCTP implementation supporting UDP encapsulation **MUST** maintain a remote UDP encapsulation port number per destination address for each SCTP association. Again, because the remote stack may be using other than the well-known port, each port may be different from each stack, but because of remapping of ports by NATs, the remote ports associated with different remote IP addresses may not be identical, even if they are associated with the same stack.

Implementation note: Because the well-known port might not be used, implementations need allow other port numbers to be specified as a local or remote UDP encapsulation port number through APIs.

5.2. Packet Format

To encapsulate an SCTP packet, a UDP header as defined in [RFC0768] is inserted between the IP header as defined in [RFC0791] and the SCTP common header as defined in [RFC4960].

Figure 1 shows the packet format of an encapsulated SCTP packet when IPv4 is used.



UDP checksum would also be counter to the goal of legacy NAT traversal.

5.4. Decapsulation Procedure

When an encapsulated packet is received, the UDP header is removed. Then the generic lookup is performed, as done by an SCTP stack whenever a packet is received, to find the association for the received SCTP packet. After finding the SCTP association (which includes checking the verification tag), the UDP source port MUST be stored as the encapsulation port for the destination address the SCTP packet is received from (see Section 5.1).

When a non-encapsulated SCTP packet is received by the SCTP stack, the encapsulation of outgoing packets belonging to the same association and the corresponding destination address MUST be disabled.

5.5. ICMP Considerations

When receiving ICMP or ICMPv6 response packets, there might not be enough bytes in the payload to identify the SCTP association which the SCTP packet triggering the ICMP or ICMPv6 packet belongs to. If a received ICMP or ICMPv6 packet can not be related to a specific SCTP association or the verification tag can't be verified, it MUST be discarded silently. This means in particular that the SCTP stack MUST NOT rely on receiving ICMP or ICMPv6 messages. Implementation constraints could prevent processing received ICMP or ICMPv6 messages.

If received ICMP or ICMPv6 messages are processed, the following mapping SHOULD apply:

1. ICMP messages with type 'Destination Unreachable' and code 'Port Unreachable' SHOULD be treated as ICMP messages with type 'Protocol Unreachable' and code 'Destination Port unreachable'. See [RFC0792] for more details.
2. ICMPv6 messages with type 'Destination Unreachable' and code 'Port unreachable' SHOULD be treated as ICMPv6 messages with type 'Parameter Problem' and code 'Unrecognized Next Header type encountered'. See [RFC4443] for more details.

5.6. Path MTU Considerations

If an SCTP endpoint starts to encapsulate the packets of a path, it MUST decrease the Path MTU of that path by the size of the UDP header. If it stops encapsulating them, the Path MTU SHOULD be increased by the size of the UDP header.

When performing Path MTU discovery as described in [RFC4820] and [RFC4821] it MUST be taken into account that one cannot rely on the feedback provided by ICMP or ICMPv6 due to the limitation laid out in Section 5.5.

If the implementation does not allow control of the don't fragment (DF)-bit contained in the IPv4 header, then Path MTU discovery can't be used. In this case, an implementation specific value should be used instead.

5.7. Handling of Embedded IP-addresses

When using UDP encapsulation for legacy NAT traversal, IP addresses that might require translation MUST NOT be put into any SCTP packet.

This means that a multi homed SCTP association is setup initially as a singled homed one and the protocol extension [RFC5061] in combination with [RFC4895] is used to add the other addresses. Only wildcard addresses are put into the SCTP packet.

When addresses are changed during the lifetime of an association [RFC5061] MUST be used with wildcard addresses only. If an SCTP endpoint receives an ABORT with the T-bit set, it MAY use this as an indication that the addresses seen by the peer might have changed.

5.8. ECN Considerations

If the implementation supports the sending and receiving of the ECN bits for the IP protocols being used by an SCTP association, the ECN bits MUST NOT be changed during sending and receiving.

6. Socket API Considerations

This section describes how the socket API defined in [RFC6458] needs to be extended to provide a way for the application to control the UDP encapsulation.

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.

6.1. Get or Set the Remote UDP Encapsulation Port Number (SCTP_REMOTE_UDP_ENCAPS_PORT)

This socket option can be used to set and retrieve the UDP encapsulation port number. This allows an endpoint to encapsulate initial packets.

```
struct sctp_udpencaps {  
    sctp_assoc_t sue_assoc_id;  
    struct sockaddr_storage sue_address;  
    uint16_t sue_port;  
};
```

sue_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 sue_assoc_id.

sue_address: This specifies which address is of interest. If a wildcard address is provided it applies only to future paths.

sue_port: The UDP port number in network byte order used as the destination port number for UDP encapsulation. Providing a value of 0 disables UDP encapsulation.

7. IANA Considerations

This document refers to the already assigned UDP port 9899 (sctp-tunneling). IANA is requested to update this assignment to refer to this document. As per [RFC6335] the Assignee should be [IESG] and the Contact should be [IETF_Chair].

Please note that the TCP port 9899 (sctp-tunneling) assignment is not needed anymore and IANA is asked to remove this TCP port number assignment.

8. Security Considerations

Encapsulating SCTP into UDP does not add any additional security considerations to the ones given in [RFC4960] and [RFC5061].

Firewalls inspecting SCTP packets must also be aware of the encapsulation and apply corresponding rules to the encapsulated packets.

An attacker might send a malicious UDP packet towards an SCTP endpoint to change the encapsulation port for a single remote address of

a particular SCTP association. However, as specified in Section 5.4, this requires the usage of one the two negotiated verification tags. This protects against blind attackers the same way as described in [RFC4960] for SCTP over IPv4 or IPv6. Non-blind attackers can affect SCTP association using the UDP encapsulation described in this document in the same way as SCTP associations not using the UDP encapsulation of SCTP described here.

9. Acknowledgments

The authors wish to thank Stewart Bryant, Dave Crocker, Gorrry Fairhurst, Tero Kivinen, Barry Leiba, Pete Resnick, Martin Stiernerling, Irene Ruengeler, and Dan Wing for their invaluable comments.

10. References

10.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S.E. and R.M. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.
- [RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, March 2007.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.
- [RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, August 2007.

- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.

10.2. Informative References

- [RFC3424] Daigle, L. IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.
- [RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, December 2011.
- [I-D.ietf-6man-udpzero]
Fairhurst, G. and M. Westerlund, "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums", draft-ietf-6man-udpzero-12 (work in progress), February 2013.
- [I-D.ietf-behave-sctpnat]
Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control Transmission Protocol (SCTP) Network Address Translation", draft-ietf-behave-sctpnat-08 (work in progress), February 2013.
- [I-D.ietf-tsvwg-natsupp]
Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control Transmission Protocol (SCTP) Network Address Translation Support", draft-ietf-tsvwg-natsupp-05 (work in progress), February 2013.

Authors' Addresses

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

Email: tuexen@fh-muenster.de

Randall R. Stewart
Adara Networks
Chapin, SC 29036
US

Email: randall@lakerest.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 7, 2014

C. Lai
W. Wang
S. Yang
T. Eckert
F. Yip
Cisco Systems
July 6, 2013

Normalization Marker for AF PHB Group in DiffServ
draft-lai-tsvwg-normalizer-02

Abstract

In DiffServ, preferential dropping of packets in AF PHB groups has long been considered beneficial, typically for video flows with discardable packets. Unfortunately, the ecosystem of bandwidth contention at congestion is very likely to discourage those video endpoints from generating packets with lower precedence markings, i.e. they would lose more packets if doing so. Thus, to offer an incentive for more collaborative and mutually beneficial behaviors of video endpoints in AF PHB groups, we propose a Normalization Marker (NM) for traffic conditioning at network edges. Deployment of NM will encourage the video endpoints to generate finer layers of intra-flow precedence (IFP) with discardable packets in more balanced distributions.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 7, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Background	6
2.1. Video Packets in Structure	6
2.2. Intra-Flow Precedence (IFP)	8
2.3. Mapping IFP to AF Markings	9
3. Normalization Marker (NM)	11
3.1. Color-Aware vs. Color-Blind Mode	12
3.2. Distribution Meter	12
3.3. Normalizer	13
4. Acknowledgements	13
5. IANA Considerations	13
6. Security Considerations	13
7. References	13
7.1. Normative References	13
7.2. Informative References	14
Authors' Addresses	14

1. Introduction

Assured Forwarding (AF) Per-Hop Behavior (PHB) groups are described in [RFC2597] (with terminology clarified in [RFC3260]) for DiffServ (DS) multimedia service classes such as realtime video conferencing and on-demand streaming. Four AF PHB groups have been defined in [RFC4594] with DS codepoint (DSCP): AF1x, AF2x, AF3x and AF4x where x=1, 2 or 3 for drop precedence in each independent AF PHB group. The DS nodes that support an AF PHB group must set configuration of Active Queue Management (AQM) properly w.r.t. those DSCP markings. For example, for AF4x PHB group which includes AF41, AF42 and AF43 markings, an AQM implementation by Weighted Random Early Detection (WRED) should be configured with some drop probabilities and queue thresholds such that the packet loss rate of AF41 \leq AF42 \leq AF43 on congestion of the queue.

For an AF PHB group, a DS boundary node or host in the DS domain should use a marking algorithm that properly assigns AF markings of drop precedence to all packets w.r.t. the traffic profiles and Service Level Agreements (SLA). For example, [RFC2697] and [RFC2698] use a token-bucket mechanism for metering each stream of packets and respectively define "srTCM" and "trTCM" markers, to mark packets by the data rate and burst size limit in traffic profiles. Those rate-control markers can be useful at DS boundary nodes for traffic conditioning [RFC2475] and to support IntServ/RSVP traffic over DS regions [RFC2998]. Multiple markers may be applied to the same stream, either on the same or multiple DS nodes along the path. For example, srTCM and trTCM can operate in a so-called "color-aware" mode such that for each incoming packet that already carries an AF marking, the local srTCM/trTCM either keeps the same or lowers the drop precedence of that packet by metering.

However, modern video codec technologies are being advanced not only in coding efficiency (i.e. better compression ratio) but also in two key areas for transport on IP networks: (1) encoder rate-control and dynamic adaptation; (2) ability to generate discardable packets in multiple layers to tolerate packet losses in the network without significant degradation of video quality observed at the decoder. For (1), the encoder dynamically limits its output rate of packets into the AF PHB group, i.e., the encoder's host is the first DS node equipped with srTCM/trTCM if it marks packets in that behavior. The next DS node is the first-hop router which may add extra srTCM/trTCM to enforce the traffic conditioning or policing from the network's perspective. Thus, we consider this an incentive for (1) because an encoder using a self rate-control is less likely to see packet losses by the network. Unfortunately, an incentive for (2) is arguably missing today.

To see the missing incentive for (2), consider the following example where 2 video flows A and B with rate control are sent in AF4x PHB group. Each sends 5Mbps on average with some burstiness, but still complies with the rate and burst limit in its traffic profile. However, A and B generate packets with AF4x markings in different distributions of percentage:

Flow A

80% or 4Mbps in AF41

20% or 1Mbps in AF42

0% or 0Mbps in AF43

Flow B

40% or 2Mbps in AF41

40% or 2Mbps in AF42

20% or 1Mbps in AF43

Flow B at above is likely using a more advanced video technology to generate multiple layers of discardable video packets, and thus, its distribution of AF4x markings looks finer and more balanced. That is, flow B acts more friendly to other flows in this AF4x PHB group.

Thus, we argue that the ecosystem in practical deployment should offer an incentive for flows to behave similarly to what flow B is doing above, i.e., on congestion, the AF4x PHB group should try to drop packets in the same amount from each flow, while a flow with finer layers of discardable packets and/or in a more balanced distribution should be able to benefit from its own efforts and see good results in video quality preservation.

Unfortunately, this incentive is still missing today. Suppose that congestion occurs in the AF4x WRED queue where A and B compete for bandwidth and there is no other flow, for simplicity. B's packet loss rate is very likely to become higher than A's, despite B's effort of acting friendly:

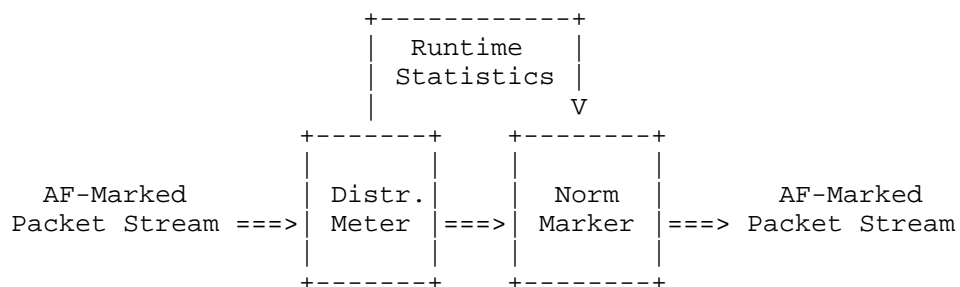
- o If the queue drops 1Mbps in total,

A sees 0% or 0Mbps loss;

B sees 20% or 1Mbps loss (all its AF43 are lost).

- o If the queue drops 4Mbps in total,
 - A sees 20% or 1Mbps loss (all its AF42 are lost);
 - B sees 60% or 3Mbps loss (all its AF42 and AF43 are lost).

Thus, to create the missing incentive at above, we propose a new "Normalization Marker" (NM) and describe it in this memo. NM can be deployed on DS boundary nodes for traffic conditioning in practical deployment with AF PHB groups for multimedia service classes. In summary, if NM is applied to a DS boundary node for an AF PHB group, it re-assigns the AF markings of all packets per flow such that the distributions of the AF markings are similar in all flows, i.e., it "normalizes" the distributions of AF markings in all flows. It also attempts to maintain the original orders of the intra-flow drop precedence carried by the input AF markings, as linearly as possible. After the AF-marking distributions are normalized, all those flows should see very similar packet loss rates at AQM for this AF PHB group on congestion of the queue. Then, a codec implementation may have better video quality preservation on network congestion if it employs a more advanced video technology to generate discardable packets with finer markings of drop precedence in a more balanced distribution.



Normalization Marker (NM) with AF PHB Group

Figure 1

Note that the use of NM is not necessarily limited to video service classes, but could be extended to wherever AF PHB groups can be used, or to any other PHB groups that require a similar incentive NM can provide.

2. Background

2.1. Video Packets in Structure

Modern video codec technologies such as ITU-T H.264/MPEG-4 AVC [H264] typically generate a stream of encoded video packets with internal structure of data dependency for decoding. This has been designed for at least 3 fundamental reasons:

- o **Coding Efficiency:** An encoder improves its coding efficiency typically by reducing spatial and temporal redundancy of the input. For video, spatial redundancy is reduced by intra-frame motion prediction and compensation, while temporal redundancy refers to inter-frame since a video stream is composed of a sequence of frames or pictures in the temporal order. With motion prediction, a frame can be encoded by referencing some pixels of the picture data that will be decoded earlier either in the same (intra) or another (inter) frame so that it can use significantly fewer bits to encode this frame. The frame where the pixels are referenced by any other frame is thus called a referenced frame in the video stream; for example, Instantaneous Decoding Refresh (IDR) in H.264 or Intra (I) frames are typically referenced by subsequential frames, while Predictive (P) frames may be referenced at the encoder's choice, by the Group of Picture (GOP) profile, and/or by some proprietary algorithm in the codec implementation.
- o **Lossy Network:** To use network transport that may lose packets, the encoder may choose to generate a stream with two or more layers each of which the packets are marked with some layer identifier (ID). The network can simply use the layer ID to determine the drop precedence of each packet in the video stream.
 - * **Layers in Hierarchy of Dependency:** If these layers are coded in hierarchy of dependency, the packets in an "enhancement" layer will depend on 1 or more "base" layers to get decoded without errors, while packets in a base layer without dependency can be independently decoded without errors.
 - + If some enhancement layer packets are lost, the decoding errors in that picture frame will not stay or cascade to other frames given that no others depend on those lost data. This nice property allows the network to safely drop packets in some enhancement layers, if needed, without badly impacting the video quality at decoder.
 - + If some base layer packets are lost, the impact can be severe since these decoding errors will stay in buffer and

cascade to all other picture pixels that depend on the lost data to decode in the current and/or a later frame. This impact can last tens of seconds as the video quality continues getting worse, resulting in unpleasant user experiences, until the decoder receives the next IDR or I frame, either on-demand or periodically, to remove those errors.

For example, H.264 Annex G defines Scalable Video Coding (SVC) using a 3-dimensional (i.e. spatical, temporal and quality) hierarchy of layer dependency at the encoder's choice, but for simplicity, it also defines a scalar number called Priority ID (PID) in its header so the network could instead use PID, if set by the encoder, to determine drop precedence in the stream.

- * Layers NOT in Hierarchy of Dependency: Sometimes the encoder will generate multiple layers without any dependency between those layers. These mechanisms usually enlarge the amount of encoded video data for vairous purposes. For example,
 - + Forward Error Correction (FEC) may be used at the encoder to generate extra FEC packets, so that the decoder can tolerate certain amounts of packet losses.
 - + Simulcast (i.e. simultaneous multicast) by an encoder will actually generate multiple layers each of which can be transmitted and decoded independently, in parallel by IP or application multicast. Each layer carries video in a different resolution and/or quality. The decoder can choose 1 or more of those layers to receive according to the required, available or detected bandwidth, packet losses, delays, jitter etc. in its network service.

With FEC and/or Simulcast, the encoder can still mark the packets with different drop precedence in those layers to better protect the more important data for video quality at decoding when congestion occurs.

- o In-Band Signaling: An encoded video stream usually carries in-band control messages that are most critical for adequate encoder and decoder behaviors. For example,
 - * H.264 Annex D defines Supplemental Enhancement Information (SEI), which could also carry proprietary codec parameters. These in-band control signals should be given the highest drop precedence.

- * Real Time Control Protocol (RTCP) carries in-band control messages for Real Time Protocol (RTP) [RFC3550], which is mostly used for realtime multimedia transmission on IP networks. RTCP messages are defined as RTP packets with special payload types in the RTP stream. RTCP packets should be given the highest drop precedence but should receive the same delay/jitter as regular RTP packets in the same stream.

2.2. Intra-Flow Precedence (IFP)

For abstraction, we define "Intra-Flow Precedence" (IFP) to represent the drop precedence in one individual flow that may carry a video stream of IP packets in multimedia networks. Here is a summary of IFP characteristics:

- o IFPs are drop precedence levels that are only significant within each individual flow.
- o IFPs are integer numbers that can be numerically compared if needed. 0 represents the highest precedence. The larger numerical value an IFP is, the lower precedence it represents.
- o The number of IFP levels in each flow is not necessarily the same.
- o IFPs between any 2 flows should NOT be compared to determine drop precedence between their packets in a queue.
- o IFPs may be assigned by the original encoder of the stream and carried in some bits field of all packets in the stream.
- o IFPs may be assigned or re-assigned by a middle box or router if it is capable of understanding the stream packet format and codec semantics.

For example, an H.264 AVC flow may have the following IFP assignments at the video encoder's choice.

IFP = 0 for in-band signals

IFP = 1 for IDR frames

IFP = 2 for referenced P (rP) frames

IFP = 3 for non-referenced P (nrP) frames and others

IFP assignments as well as their distribution can vary a lot among different encoder implementations and codec profiles. For example, some encoders may generate both long-term and short-term referenced P

frames, where a long-term referenced P frame should have higher drop precedence. In case of H.264 SVC, the IFP assignments could simply be the same as the PID assignments if set by the encoder properly, or be calculated based on the SVC layer ID that has 3 tuples for the spatial, temporal and quality dimensions, respectively.

2.3. Mapping IFP to AF Markings

When a flow is sent in an AF PHB group, the number of its IFP levels is not necessarily equal to the number of the AF markings. In fact, since each of the currently defined AF PHB groups has only 3 AF markings, it is likely that an encoder or DS node needs to apply an n-to-1 mapping from IFPs to AF markings in practice.

The mapping decision is made usually by the encoder, but can also be made by another DS node if necessary and if the DS node is able to understand the encoded video packets, which may require Deep Packet Inspection (DPI), e.g. to read in RTP payload and parse the H.264 headers [RFC6184], or in a proprietary bits field in the IP payload, to retrieve or calculate the IFP of each packet in a flow before locally mapping the IFP to an AF marking.

This n-to-1 mapping can be arbitrary but should be appropriate. Consider 2 IFPs, say x and y , where x and y are mapped to AF markings $AF(x)$ and $AF(y)$, respectively. Then, the mapping should ideally obey the following criteria to keep linearity from IFPs to AF markings.

If $x < y$, $AF(x) \leq AF(y)$;

If $x > y$, $AF(x) \geq AF(y)$.

Although the above two do NOT imply that if $x = y$, $AF(x) = AF(y)$, it is usually so in practical implementation as it is straightforward. Then, if the encoder algorithm generates a lot of packets with the same IFP, all those packets will be assigned the same AF marking, possibly resulting in an unbalanced distribution of AF markings in the AF PHB group. Thus, an encoder with advanced technologies should make good efforts to generate packets with a finer and more balanced IFP distribution in the first place.

For example, if AF4x PHB group is used to send an H.264 AVC flow with the IFP assignments in the example of Section 2.2, one possible IFP-to-AF4x mapping is:

$AF(0) = AF41$

$AF(1) = AF41$

AF(2) = AF42

AF(3) = AF43

This mapping actually results in the following AF markings:

AF41 for in-band signals and IDR frames

AF42 for referenced P (rP) frames

AF43 for non-referenced P (nrP) frames and others

Now, consider two encoders that generate flow A and B, respectively, both using this mapping, but with different IFP distributions as follows.

Flow A

5% in IFP=0 for in-band signals

75% in IFP=1 for IDR frames

20% in IFP=2 for rP frames

Flow B

5% in IFP=0 for in-band signals

35% in IFP=1 for IDR frames

40% in IFP=2 for rP frames

20% in IFP=3 for nrP frames

Thus,

Flow A

80% in AF41

20% in AF42

0% in AF43

Flow B

40% in AF41

40% in AF42

20% in AF43

This results in exactly the two AF marking distributions that we have previously used in Section 1.

Note that in terms of encoded data size, an IDR frame is typically 10 times larger than a P frame on average. Assume that flow B's coding efficiency has rP twice as large as nrP. Then, flow A and B might be sending frames periodically in patterns by Group of Picture (GOP) as follows:

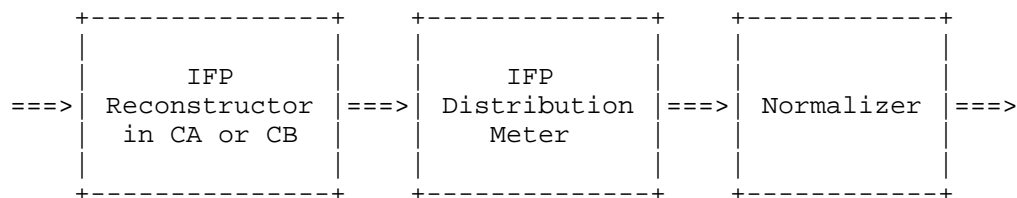
Flow A: IDR, rP, rP, rP

Flow B: IDR, rP, nrP, rP, nrP, rP, nrP, rP, nrP

If so, it shows that flow B's encoder is making efforts to generate discardable packets with more layers in a more balanced distribution, which is desirable.

3. Normalization Marker (NM)

Referring to Figure 2, NM has 3 major components: IFP reconstructor, IFP distribution meter, and normalizer. NM may operate in either "color-aware" (CA) or "color-blind" (CB) mode.



Normalization Marker (NM) Architecture

Figure 2

The packets arrive at the IFP reconstructor which determines the IFP of each packet depending on whether NM is in CA or CB mode. This is fed into the IFP distribution meter that keeps a runtime statistics. Then, by the runtime statistics and the IFP of the very packet, the normalizer writes a proper AF-marking in that packet.

3.1. Color-Aware vs. Color-Blind Mode

When NM operates in "color-aware" (CA) mode, it reads the incoming AF-markings that are carried in the packets as the drop precedence. This CA mode should be supported in all NM implementations.

When NM operates in "color-blind" (CB) mode, which is optionally supported, it reads certain bits field(s) other than the AF-markings in the packets to determine the actual drop precedence of that packet. This implies that NM may need DPI in the packets, e.g. parsing into H.264 AVC header in each RTP packets, or alternatively use some method where the drop precedence is carried from the encoder in a customized bits field other than the AF-marking in each packet.

In comparison, CB is more complex than CA in implementation. However, CB could probably produce better normalization results because the AF-markings are actually outcomes of an n-to-1 mapping from IFPs, as previously mentioned in Section 2.3, which can reduce granularity, e.g. for IFPs x and y , if $x > y$ at encoder, it is possible that $AF(x) = AF(y)$ when NM sees those packets in CA mode. On the contrary, NM in CB mode may reconstruct IFPs $x > y$ for those packets by local DPI.

Note that NM in CB mode may fail to determine the IFP of a packet for various reasons at runtime. If so, NM should randomly assign an IFP to each of those packets with an even distribution over the IFPs. The failure could be due to payload encryption that prevents DPI. Another reason may be that the NM does not support the codec used for encoding those packets in the flow. For example, an NM might only support H.264 AVC but is unable to parse packets in H.264 Annex G (SVC), so it fails to determine the IFPs of packets in an H.264 SVC flow.

3.2. Distribution Meter

The IFP distribution meter keeps a runtime statistics of the IFPs per flow so that the normalizer will be able to assign a proper AF-marking for each packet. The types of statistics to collect at runtime depend on the NM algorithm in the implementation.

For example, an NM implementation may keep a counter of packets per IFP in a flow since the beginning of the flow's lifetime. Another implementation may choose to keep only the running average of the packet counter per IFP. An even simpler implementation may choose to keep only the running average of IFPs of all packets per flow.

3.3. Normalizer

The normalizer should reference the runtime statistics kept by the IFP distribution meter, and adaptively map the IFP of the very packet to an AF marking, such that the resulting AF-marking distributions for all flows are similar or even identical to a target distribution.

The target distribution of an NM can be simply an even distribution over all possible AF-markings in the AF PHB group. However, in a more complex NM implementation, it may allow configuration for other target distributions as appropriate with the AQM configuration.

4. Acknowledgements

The authors would like to thank many colleagues for comments and supports, and thank Shuai Dai for testing NM with actual H.264 video endpoints.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

This memo has no security consideration at the time of writing.

7. References

7.1. Normative References

- [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.
- [RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.
- [RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, September 1999.
- [RFC2698] Heinanen, J. and R. Guerin, "A Two Rate Three Color Marker", RFC 2698, September 1999.
- [RFC2998] Bernet, Y., Ford, P., Yavatkar, R., Baker, F., Zhang, L.,

Speer, M., Braden, R., Davie, B., Wroclawski, J., and E. Felstaine, "A Framework for Integrated Services Operation over Diffserv Networks", RFC 2998, November 2000.

[RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, April 2002.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, August 2006.

7.2. Informative References

[H264] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services", March 2010.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC6184] Wang, Y., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, May 2011.

Authors' Addresses

Cheng-Jia Lai
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: chelai@cisco.com

Wenyi Wang
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: wenywang@cisco.com

Stan Yang
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: stanyang@cisco.com

Toerless Eckert
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: eckert@cisco.com

Fred Yip
Cisco Systems
San Diego, CA
US

Email: fyip@cisco.com

Network WG
Internet-Draft
Intended status: Proposed Standard
Expires: January 4, 2015
Updates: RFC 2205 & 4495 (if published as an RFC)

James Polk
Subha Dhesikan
Cisco
July 4, 2014

Resource Reservation Protocol Multiple Instance Object
draft-polk-rsvp-multi-instance-object-02.txt

Abstract

This document creates the framework for a new Resource Reservation Protocol version 1 (RSVP) object for instances in which there are multiple occurrences of existing RSVP objects is to be included within the same RSVP message. This document offers two instances for multiple versions of the same object will be valid in RSVP messages, for more than one traffic specification object (TSPEC), and more than one TSPEC priority object (PREEMPTION_PRI).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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

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

This Internet-Draft will expire on January 4, 2015.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Terminology	5
3. Framework for the MULTI_INSTANCE Object	6
3.1 The MULTI_INSTANCE Object Format	9
3.2 Rules for building a MULTI_INSTANCE Object	9
4. Multiple TSPEC Objects in the MULTI_INSTANCE Object	10
5. Multiple PREEMPTION_PRI Elements in the MULTI_INSTANCE Object	12
6. IANA Considerations	15
7. Security Considerations	16
8. Contributing Authors	16
9. Acknowledgements	17
10. References	17
10.1 Normative References	17
10.2 Informative References	18
Author's Addresses	18
Appendix	18

1. Introduction

This document creates the framework for a new Resource Reservation Protocol version 1 (RSVP) [RFC2205] object for instances in which there is a need to carry multiple occurrences of included RSVP object within the same RSVP message. The need for multiple versions of existing objects is for environments in which the information conveyed within these objects may or may not be grantable by the network. To optimization this operation, if a different version of the same object, with different information or demands, can be included without the need for that rejecting entire RSVP message. For example, the initial RSVP PATH message contains a request for a 12Mbps bandwidth reservation, but that amount is not grantable by one or more network nodes. If a reduced amount of bandwidth can still be granted, and is acceptable to the network as well as both endpoints, allowing that PATH message to contain a backup bandwidth request for, say 4Mbps, saves the time of completely rejecting the initial PATH and having the sender generate a new PATH. A complete rejection to this scenario is how RSVP operates today.

This is a general purpose optimization for RSVP, and will allow any RSVP object to have multiple versions of an existing object, provided that existing object is specified to do so. It is important to understand that RSVP operates normally, with all Objects and elements in their native locations. This document offers two instances for multiple versions of the same object will be valid in RSVP messages, for more than one traffic specification object (TSPEC) [RFC2210], and more than one priority element (PREEMPTION_PRI) [RFC3181]. This extension will bring RSVP more in line with existing application layer protocols that offer multiple choices for the specifics within a call or session. At the same

time, all extra versions of Objects or elements are contained in a single location that is ignored if not understood. Thus, backwards compatibility is assured.

Realtime session set-up protocols such as SIP [RFC3261] carry a Session Description Protocol (SDP) [RFC4566] payload which establishes the parameters for rich media calls (i.e., voice, video) between two or more endpoints. Since the late 1990s, SDP has had the capability to offer more than one codec per application type (i.e., more than one audio payload type and/or more than one video payload type), which can be carried in the same session set-up message. This means a calling endpoint can give the called party a list of codecs to choose from for that call, as well as multiple applications for that call.

With this RSVP extension, for example, a SIP voice and/or video call can have a reservation adapt to whichever codec(s) are picked for that call, without wasting unnecessary bandwidth that will not be utilized.

Visually, Figure 1. is a normal RSVP reservation set-up exchange that is accepted by all RSVP enabled nodes.

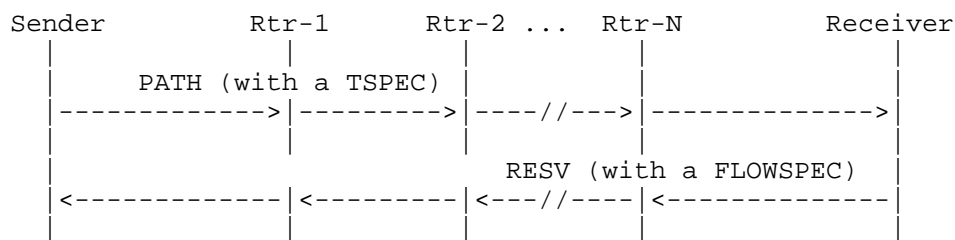


Figure 1. Concept of RSVP in a Single Direction

However, Figure 2. is a normal RSVP reservation set-up exchange that is rejected as the reservation is partially established. We will use bandwidth as the reason for the rejection because it is probably the easiest thing to understand about RSVP, that it creates reservation of fixed bandwidth.

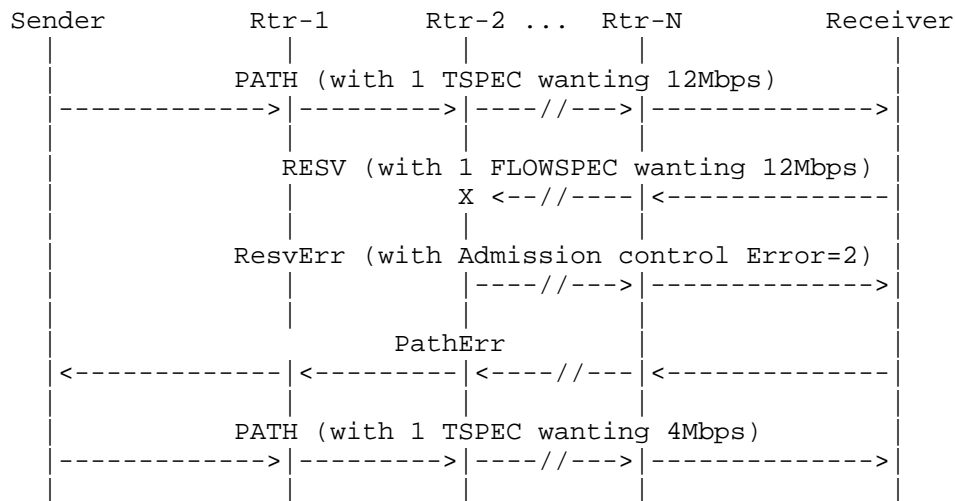


Figure 2. Concept of RSVP Rejection due to Limited Bandwidth

Rtr-2 in the above example reservation attempt rejects the bandwidth requested for this reservation. Once the Sender receives the PathErr message indicating why the rejection occurred, it can attempt a new reservation requesting less bandwidth. Regrettably, this is a bit of a guessing game put on the Sender to figure out how much bandwidth to request next. When this scenario is complicated when the reservation request is initiated because of a layer 7 signaling protocol, such as SIP, to establish a call between two endpoints, as defined in [RFC3312]. All the users experience is further delay as RSVP attempts to successfully establish the reservation before "the phone can ring".

Presently, translating a (SIP) layer 7 operation into RSVP at layer 4, only a single reservation can be established per application (i.e., one for voice, one for video) at a time (without creating chaos). This one reservation request would most probably its bandwidth request using the codec with the largest bandwidth requirements. Bandwidth parameters are conveyed within a traffic specification (TSPEC), as defined in [RFC2210]. Once one considers the bandwidth needs of present day video codecs, always initially setting up the maximum bandwidth reservation is less than optimal (some might argue criminal).

If, on the other hand, the initial RSVP PATH message could contain more than one version of a TSPEC, say one per codec. Then the reservation would be established with the greatest amount of bandwidth the network could grant at its most congested node in the signaling path, which would in turn choose for the endpoints which codec within SDP is selected for this call.

Thus, this extension would solve the problem in Figure 2. in this way,

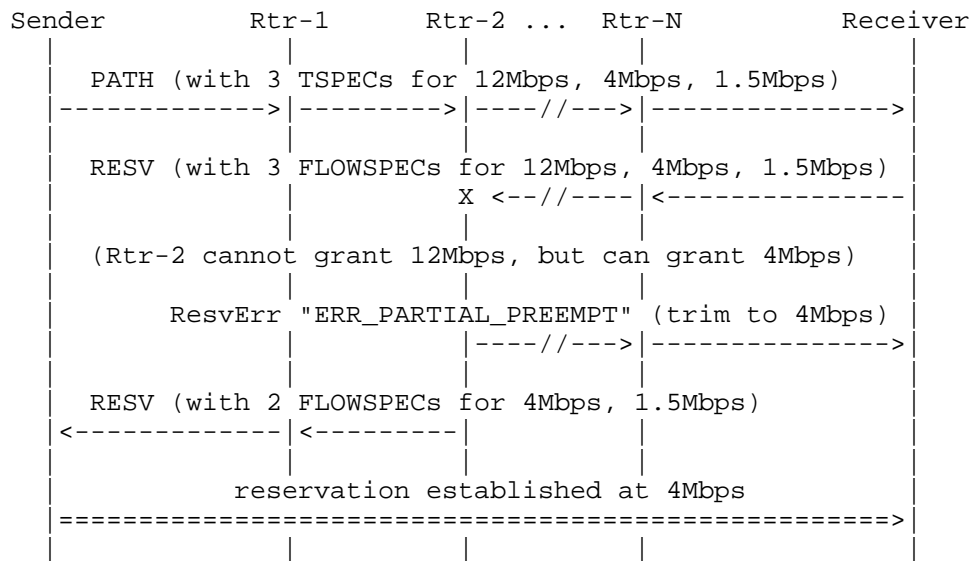


Figure 3. Concept of RSVP Rejection due to Limited Bandwidth

Figure 3. shows a RSVP PATH message containing 3 TSPECs (12Mbps, 4Mbps, and 1.5Mbps), placed in that order in the PATH. Router 2 (Rtr-2) cannot grant the RESV message at 12Mbps, but can grant the 4Mbps bandwidth request. Rtr-2 trims the bandwidth upstream with a slight modification to the procedures defined in RFC 4495, and transmits the RESV downstream without the 12Mbps bandwidth request, which was removed from the RESV. The Sender, in this example, receives the RESV with 4Mbps and the reservation is established.

In Section 3., we will create the framework and format for the MULTI_INSTANCE Object. In Section 4., we will show how to include multiple TSPEC Objects within this MULTI_INSTANCE Object, as well as stipulate the rules for TSPEC usage. In Section 5., we will show how to include multiple PREEMPTION_PRI Objects within this MULTI_INSTANCE Object, as well as stipulate the rules for PREEMPTION_PRI usage. Section 6. will have the IANA registry considerations, and Section 7. will have the Security considerations.

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] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

3. Framework for the MULTI_INSTANCE Object

The format of all RSVP Objects is based on a series of 32-bit words. This is true with the MULTI_INSTANCE Object as well. Normally, RSVP and IntServ documents specify Objects in a 32-bit wide, top down format, where the most significant bit is the top left bit, and the least significant bit on the right. This is loosely shown in Figure 4. below.

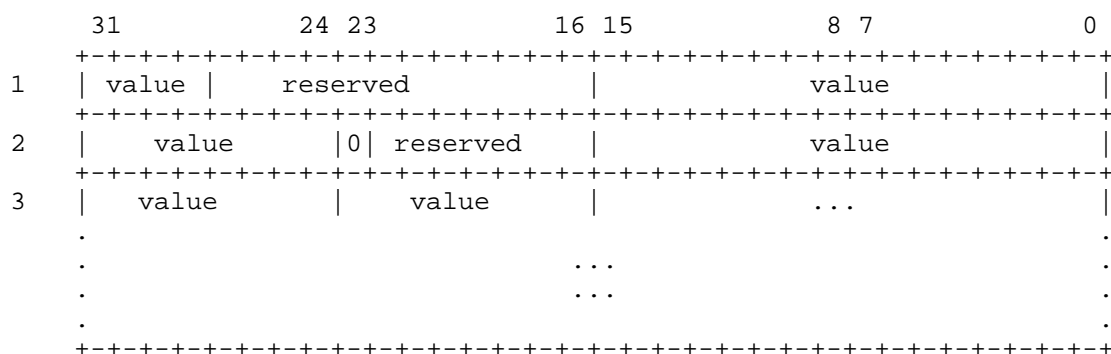


Figure 4. Generic RSVP Format for Illustration Purposes

The individual field value lengths within each RSVP Object depend on the Object, thus Figure 4. is merely an example (which happens to be the first 3 words format of a TSPEC).

However, RSVP messages can be quite long in this format, so what one usually sees in documents are each individual Object and no overall RSVP message format. Each Object has an field identifier indicating which RSVP message this Object is within (e.g., PATH, RESV, REFRESH), as well as a 'Parameter ID' indicating which Object within this (e.g., TSPEC, Rspec, Policy_Data).

Looking at RSVP another way, to illustrate the point about where certain parts can be within an overall RSVP message, Figure 5. Shows an example RSVP message on its side, where the top of the message is on the left and the bottom of the message is on the right. With that in mind, the most significant bit of the top 32-bit word is on the lower left of Figure 5., and the least significant bit is on the upper left. The length of this message can vary and does not represent anything other than this message has some size to it; i.e., it has a number of Objects within this message, including a sender_descriptor where the 'primary' TSPEC Object resides, and Policy_Data Object where the PREEMPTION_PRI Object resides. Additionally in the RSVP message is the proposed MULTI_INSTANCE Object, which is neither in the sender_descriptor or Policy_Data Object.

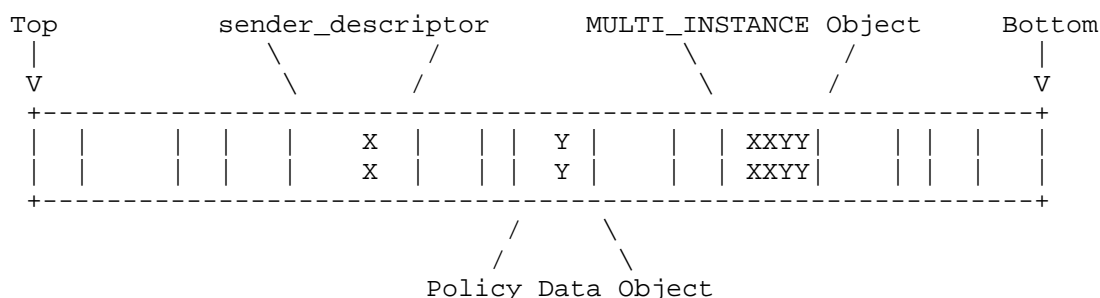


Figure 5. Generic RSVP Format Shown Sideways

It is important to remember that RSVP enabled nodes will always ignore Objects that are not understood. This allows the protocol to be extended before all the RSVP nodes are upgraded to understand new functions and capabilities. In other words, no one expects a single 'flag day' upgrade to occur in all routers at the same time in the same network, which could be disruptive if not performed correctly.

An important aspect of this new Object is that the initial copy or instance of the Object, however many Objects have multiple instances in this RSVP message, MUST remain in its original place within the message. We will refer to this original version of an Object or element to be the 'primary' version or copy. Its placement allows RSVP to operate normally. The MULTI_INSTANCE Object only carries a second, third, etc. versions of Objects. Once the RSVP node determines that it cannot grant what is asked for in an existing Object, it will look to the MULTI_INSTANCE Object for the next instance of that Object to replace the original with. Failing this, the RSVP message will mostly likely be rejected through the normal procedures already defined in RSVP documentation.

To give a practical example of this, we will use the message flow from Figure 3. In it, we have a PATH message carrying not one, but three TSPECs for 12Mbps, 4Mbps and 1.5Mbps. Once Rtr-2 cannot grant the primary TSPEC asking for 12Mbps, that router discards that TSPEC, from the RSVP message. It knows to look into the MULTI_INSTANCE Object for a second version of the TSPEC. Finding two (4Mbps and 1.5Mbps), Rtr-2 moves the 4Mbps TSPEC completely into the sender_descriptor as the new 'primary' TSPEC and attempts to establish the reservation at 4Mbps. In this case, 4Mbps is granted and transmits the RESV upstream towards Rtr-1 with only one remaining TSPEC in the MULTI_INSTANCE Object.

[EDITOR'S NOTE: It is important to state that, so far, we have not defined where this MULTI_INSTANCE Object goes within the RSVP message.]

The MULTI_INSTANCE Object can have a number of versions of the same Object that is within the RSVP message, as well as can have more than one different type of Object. To this end, here is the proposed

generic format for the MULTI_INSTANCE Object that is only carrying a single other Object

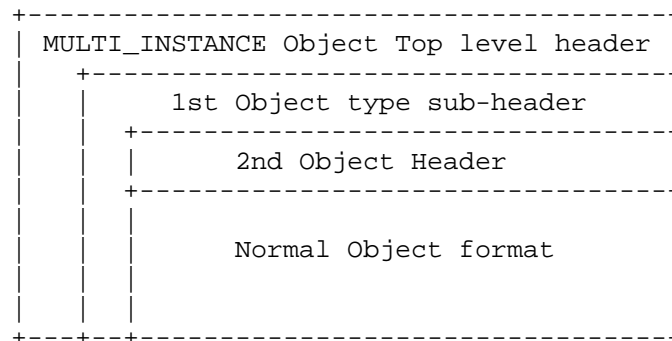


Figure 6. MULTI_INSTANCE with 1 Object

Figure 6. shows a complete second version of an existing RSVP Object, which can be removed and copied bit-for-bit into the normal placement of this Object within the RSVP message. It is important to note that this MUST be a complete new copy of a valid Object.

Figure 7. shows a MULTI_INSTANCE Object with a second and third version of the same RSVP Object

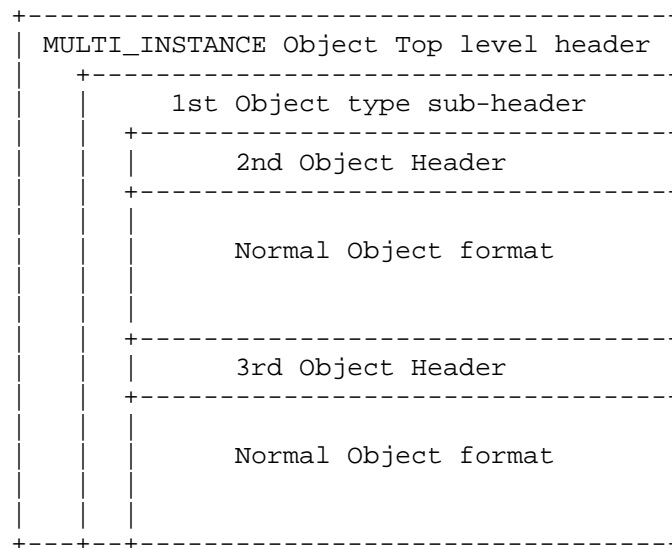


Figure 7. MULTI_INSTANCE with 2 Objects

Again, version 2 and 3 are completely valid versions of the RSVP Object they are meant to replace, with no change in any value allowed.

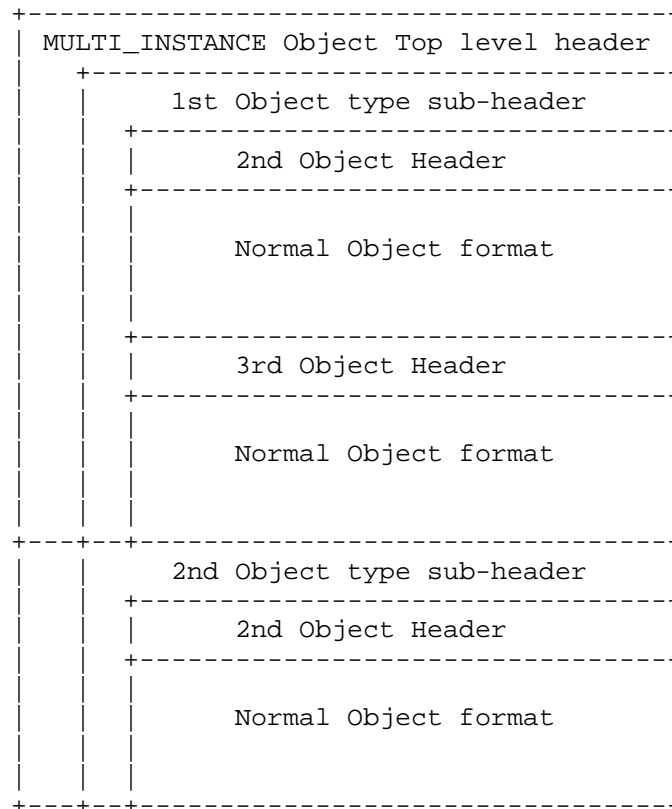


Figure 8. MULTI_INSTANCE with 2 Objects

Figure 8. adds a second type of Object to the MULTI_INSTANCE Object that is shown in Figure 7. To be able to add another type of Object, and not a second copy of the same Object, a new Object Type header is REQUIRED to preface the first 32-bit word of the new Object. These two different Objects carried within the MULTI_INSTANCE Object can be related, or they might not have anything to do with each other.

3.1 The MULTI_INSTANCE Object Format

The multi-32-bit word format of the MULTI_INSTANCE Object is TBD in a subsequent revision of this document.

3.2 Rules for building a MULTI_INSTANCE Object

The following are the rules for implementations of the MULTI_INSTANCE object:

- #1 - having only 1 *SPEC or Object is allowed in the MULTI_INSTANCE

Object (i.e., a grouping can have a single entry)

- #2 - more than one *SPEC or Object is allowed in the MULTI_INSTANCE Object (i.e., separate groups can have a single entry each)
- #3 - more than one *SPEC or Object is allowed in the MULTI_INSTANCE Object (i.e., separate groups can have a multiple entries each)
- #4 - some groupings within MULTI_INSTANCE MUST be paired in whenever a single instance occurs in any group.

In other words, based on rule #3, if a TSPEC is in each group, so MUST there be an RSPEC if any RSPEC is within this MULTI_INSTANCE Object. An RSPEC is an example of a *SPEC that MUST NOT be alone without its TSPEC.

4. Multiple TSPEC Objects in the MULTI_INSTANCE Object

This document defines the framework for the MULTI_INSTANCE Object, as well as for two Objects to be available for inclusion within this new Object: the TSPEC Object and the PREEMPTION_PRI Object (detailed in Section 5.). This section deals with how to include one or more TSPEC Objects within the MULTI_INSTANCE Object.

This document specifies if the reservation is to be Controlled Load [RFC2211], the entire TSPEC, including the two 32-bit word headers (totaling eight 32-bit words), are included in the MULTI_INSTANCE object. An example of a TSPEC from RFC 2210 is here in Figure 9.:

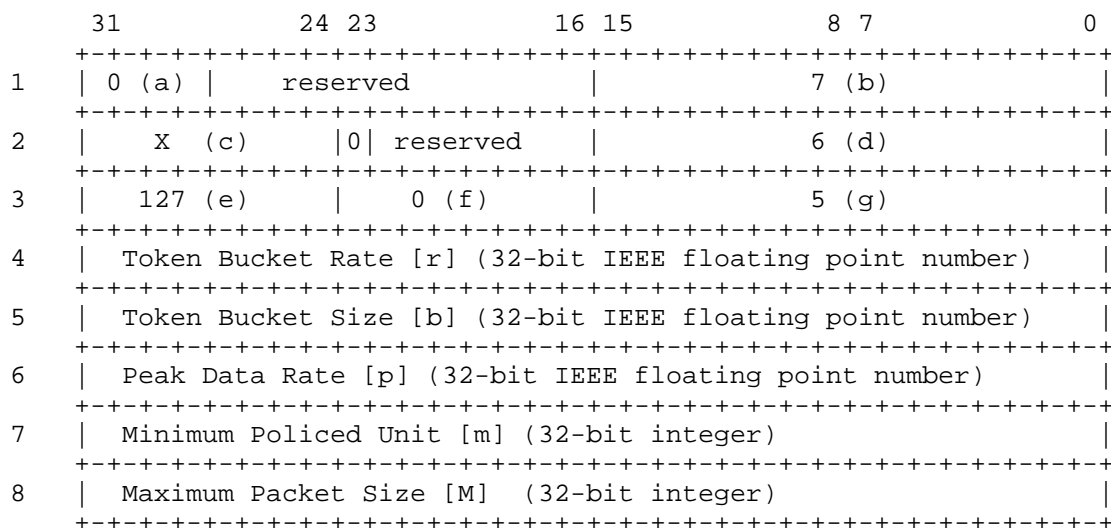


Figure 9. Controlled Load SENDER_TSPEC in a PATH

- (a) - Message format version number (0)
- (b) - Overall length (7 words not including header)
- (c) - Service header, service number
 - '1' (Generic information) if in a PATH message;
- (d) - Length of service data, 6 words not including per-service header
- (e) - Parameter ID, parameter 127 (Token Bucket TSPEC)
- (f) - Parameter 127 flags (none set)
- (g) - Parameter 127 length, 5 words not including per-service header

This document specifies if the reservation is to be Guaranteed Service, the entire TSPEC and RSPEC, including the two 32-bit word headers (totaling eleven 32-bit words), are included in the MULTI_INSTANCE object as a single consecutive chunk.

A request guaranteed service reservation contains a TSPEC and RSPEC [RFC2215], as shown in Figure 10.:

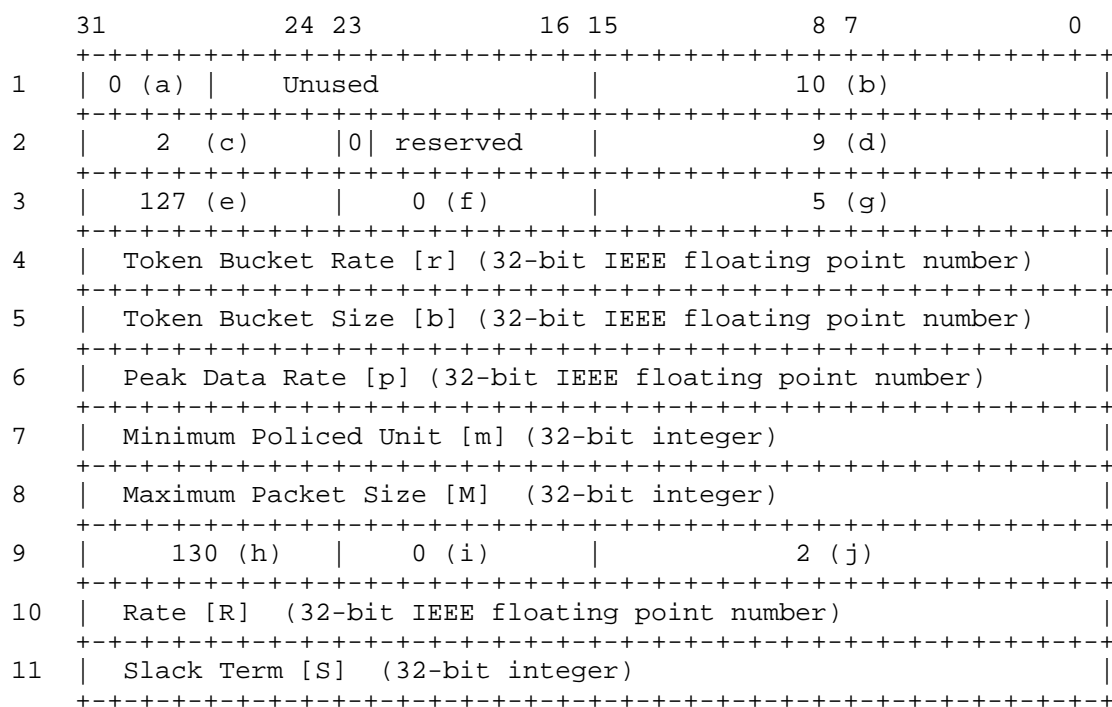


Figure 10. Guaranteed Service SENDER_TSPEC in a PATH

- (a) - Message format version number (0)
- (b) - Overall length (9 words not including header)
- (c) - Service header, service number 2 (Guaranteed)
- (d) - Length of per-service data, 9 words not including per-service header

- (e) - Parameter ID, parameter 127 (Token Bucket TSpec)
- (f) - Parameter 127 flags (none set)
- (g) - Parameter 127 length, 5 words not including parameter header
- (h) - Parameter ID, parameter 130 (Guaranteed Service RSpec)
- (i) - Parameter xxx flags (none set)
- (j) - Parameter xxx length, 2 words not including parameter header

The difference in structure between the Controlled-Load FLOWSPEC and Guaranteed FLOWSPEC is the RSPEC, defined in [RFC2212]. The difference with respect to the MULTI_INSTANCE Object is found in the first 32-bit word, value 'b' above - the TSpec Object overall length. This will tell a node whether it is a Controlled Load or Guaranteed Service TSpec.

As a reminder, TSPECs contained in the MULTI_INSTANCE Object MUST NOT be altered when moved from the MULTI_INSTANCE Object to the sender_descriptor or FLOWSPEC. Generically, this needs to be a simple cut and paste operation.

If there are multiple TSPECs in the MULTI_INSTANCE Object, each MUST be the same type of TSpec. In other words, there MUST NOT be a mix of Controlled Load with Guaranteed Service TSPECs in the same MULTI_INSTANCE Object.

RFC 4495 defines how existing reservations can partially preemption (trim) the agreed upon bandwidth assigned to an existing reservation. This specification extends RFC 4495 by allowing that trimming of bandwidth assigned to a reservation to occur during reservation establishment downstream. This occurs when a node upstream cannot grant the bandwidth already granted downstream, but that upstream node can grant a reduced amount of bandwidth from another TSpec within FLOWSPEC, from within the MULTI_INSTANCE Object. This operation is shown in Figure 3.

5. Multiple PREEMPTION_PRI Elements in the MULTI_INSTANCE Object

The order of the TSPECs within the MULTI_INSTANCE Object is one way to determine which is the next TSpec to be processed by a router. Another way of determining which TSpec is the next one to be processed is by allowing the dynamic bandwidth selection to reflect a different reservation priority for each of the multiple "bandwidth" associated with a reservation.

[RFC2750] presents a set of extensions for supporting generic policy based admission control in RSVP. These extensions include the standard format of POLICY_DATA objects, and a description of RSVP's handling of policy events. These extensions are consistent with the framework for policy-based admission control presented in [RFC2753]. POLICY_DATA objects are carried by RSVP messages and contain policy information. The exchange of POLICY_DATA objects between policy-capable nodes along the data path, supports the generation

and enforcement of consistent end-to-end admission control policies.

POLICY_DATA objects contain a list of Policy Elements that each contain a single unit of information necessary for the evaluation of policy rules. Multiple policy elements are already specified. For example, [RFC2872] specifies the Application and Sub Application Identity policy element for use with RSVP.

[RFC3181] specifies another policy element, the Preemption Priority Policy Element, that can be signaled in RSVP so that network node may take into account this policy element in order to preempt some previously admitted low priority sessions in order to make room for a newer, higher priority session. The Preemption Priority Policy Element (PREEMPTION_PRI) contains:

- o one Preemption Priority specifying the priority of the new flow compared with the defending priority of previously admitted flows.
- o one Defending Priority that is used once this reservation is established to compare with the preemption priority of new flows.

The format of preemption priority policy element (copied from RFC 3181) is as follows:

+-----+-----+-----+-----+			
Length (12)		P-Type = PREEMPTION_PRI	
+-----+-----+-----+-----+			
Flags	M. Strategy	Error Code	Reserved(0)
+-----+-----+-----+-----+			
Preemption Priority		Defending Priority	
+-----+-----+-----+-----+			

Figure 11. Preemption Priority Policy Element Format

Length: 16 bits

Always 12. The overall length of the policy element, in bytes.

P-Type: 16 bits

PREEMPTION_PRI = 1

This value is registered with IANA, see Section 7.

Flags: 8 bits

Reserved (always 0).

Merge Strategy: 8 bit

- 1 Take priority of highest QoS: recommended
- 2 Take highest priority: aggressive
- 3 Force Error on heterogeneous merge

Reserved: 8 bits

Error code: 8 bits

- | | | |
|---|---------------|--|
| 0 | NO_ERROR | Value used for regular PREEMPTION_PRI elements |
| 1 | PREEMPTION | This previously admitted flow was preempted |
| 2 | HETEROGENEOUS | This element encountered heterogeneous merge |

Reserved: 8 bits

Always 0.

Preemption Priority: 16 bit (unsigned)

The priority of the new flow compared with the defending priority of previously admitted flows. Higher values represent higher Priority.

Defending Priority: 16 bits (unsigned)

Once a flow was admitted, the preemption priority becomes irrelevant. Instead, its defending priority is used to compare with the preemption priority of new flows.

For any specific flow, its preemption priority MUST always be less than or equal to the defending priority.

The preemption priority and defending priority of the Preemption Priority Policy Element carried in a RESV message MUST be associated with the flow specification carried in the FLOWSPEC object. There MUST be either no Preemption Priority Policy Element carried in the MULTI_INSTANCE Object, or there needs to be the exact same number of Preemption Priority Policy Element as there are TSPEC Objects. This MUST be a one-to-one mapping of numbers. For example, the preemption priority and defending priority of the first (respectively second) sub-element (when present) of the MULTI_INSTANCE Object is to be associated with the first (respectively second) flow specification (when present) in the MULTI_INSTANCE Object.

If a RESV message contains a dissimilar number of TSPECs than Preemption Priority Policy Elements in the MULTI_INSTANCE object, but contains a Preemption Priority Policy Element in the POLICY_DATA object, then the Preemption Priority Policy Element in the MULTI_INSTANCE object MUST be ignored, and all TSPECs retain the priority properties of the Preemption Priority Policy Element in the POLICY_DATA object.

An example MULTI_INSTANCE Object with 2 TSPEC Objects and 2 Preemption Priority Policy Element is showing generically in Figure 12.

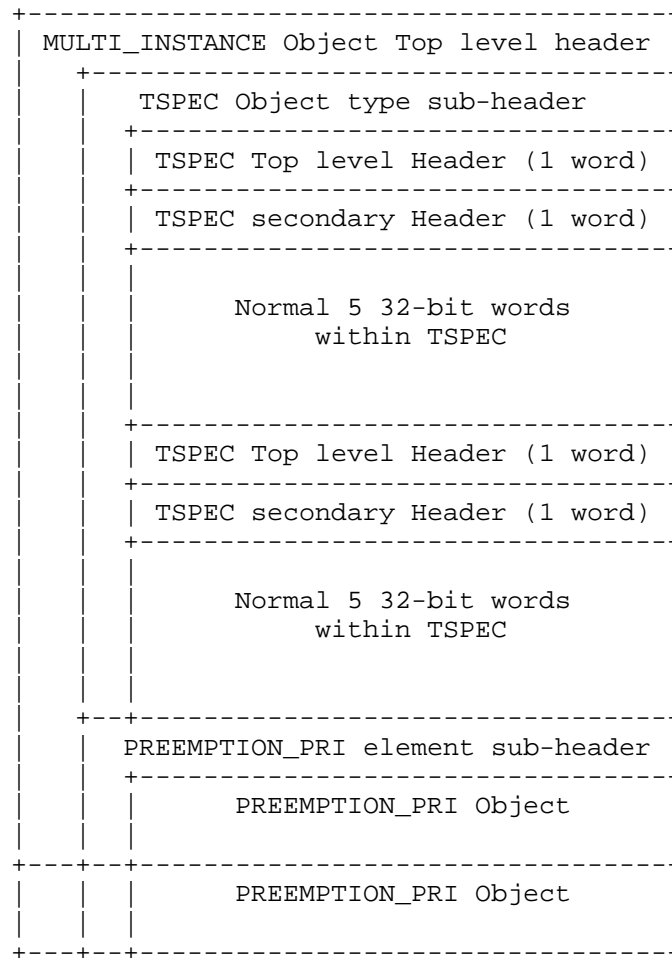


Figure 12. MULTI_INSTANCE with 2 TSPECs and 2 PREEMPTION_PRIs

6. IANA Considerations

This document IANA registers the following new parameter name in the rsvp-parameters assignments at [IANA]:

Registry Name: Parameter Names

Registry:

Value	Description	Reference
125	Multiple_Instance_object	[RFCXXXX]

Where RFCXXXX is replaced with the RFC number assigned to this Document.

This document IANA registers the following new error subcode in the

Error code section, under the Admission Control Failure (error=1), of the rsvp-parameters assignments at [IANA]:

Registry Name: Error Codes and Globally-Defined Error Value
Sub-Codes

Registry:

"Admission Control
Failure"

Error Subcode	meaning	Reference
6	= MULTI_INSTANCE bandwidth unavailable	[RFCXXXX]

7. Security Considerations

The security considerations for this document do not exceed what is already in RFC 2205 (RSVP), as nothing in either of those documents prevent a node from requesting a lot of bandwidth in a single TSPEC, or what priority values are given in a Preemption Priority Policy Element. This document merely reduces the signaling traffic load on the network by allowing many requests that fall under the same policy controls to be included in a single round-trip message exchange.

Further, this document does not increase the security risk(s) to that defined in RFC 4495, where this document creates additional meaning to the RFC 4495 created error code 102.

A misbehaving Sender can include too many TSPECs in the MULTI_INSTANCE object, which can lead to an amplification attack. That said, a bad implementation can create a reservation for each TSPEC received from within the RESV message. The number of TSPECs in the new MULTI_INSTANCE object is limited, and the spec clearly states that only a single reservation is to be set up per RESV message.

To ensure the integrity of RSVP, the RSVP Authentication mechanisms defined in [RFC2747] and [RFC3097] SHOULD be used. Those protect RSVP message integrity hop-by-hop and provide node authentication as well as replay protection, thereby protecting against corruption and spoofing of RSVP messages.

8. Contributing Authors

The authors here would like to thank the authors of draft-lefaucheur-tsvwg-rsvp-multiple-preemption-02 for allowing that draft to be merged with this draft, specifically for the Preemption Priority Policy Element discussion in Section 5. They are:

Francois Le Faucheur
Arun Kudur and
Ashok Narayanan

9. Acknowledgements

The authors wish to thank Fred Baker, Joe Touch, Bruce Davie, Dave Oran, Ashok Narayanan, Lou Berger, Lars Eggert, Arun Kudur, Janet Gunn and Ken Carlberg for their helpful comments and guidance in this effort.

10. References

10.1 Normative References

- [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997
- [RFC2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997
- [RFC2210] J. Wroclawski, "The Use of RSVP with IETF Integrated Services", RFC 2210, September 1997
- [RFC2211] J. Wroclawski, "Specification of the Controlled-Load Network Element Service ", RFC 2211, September 1997
- [RFC2212] S. Shenker, C. Partridge, R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, September 1997
- [RFC2215] S. Shenker, J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements", RFC 2212, September 1997
- [RFC2747] F. Baker, B. Lindell, M. Talwar, " RSVP Cryptographic Authentication", RFC2747, January 2000
- [RFC2750] S. Herzog, "RSVP Extensions for Policy Control", RFC 2750, January 2000
- [RFC2753] R. Yavatkar, D. Pendarakis, R. Guerin, "A Framework for Policy-based Admission Control", RFC 2753, January 2000
- [RFC2872] Y. Bernet, R. Pabbati, "Application and Sub Application Identity Policy Element for Use with RSVP", RFC 2872, June 2000
- [RFC3097] R. Braden, L. Zhang, "RSVP Cryptographic Authentication -- Updated Message Type Value", RFC 3097, April 2001
- [RFC3181] S. Herzog, "Signaled Preemption Priority Policy Element", RFC 3181, October 2001

- [RFC3261] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, May 2002.
- [RFC3312] G. Camarillo, Ed., W. Marshall, Ed., J. Rosenberg, "Integration of Resource Management and Session Initiation Protocol (SIP)", RFC 3312 Preconditions, October 2002
- [RFC4495] J. Polk, S. Dhesikan, "A Resource Reservation Protocol (RSVP) Extension for the Reduction of Bandwidth of a Reservation Flow", RFC 4495, May 2006
- [RFC4566] M. Handley, V. Jacobson, C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006

10.2 Informative References

draft-lefaucheur-tsvwg-rsvp-multiple-preemption

draft-ietf-tsvwg-intserv-multiple-tspec

Author's Addresses

James Polk
3913 Treemont Circle
Colleyville, Texas, USA
+1.817.271.3552

mailto: jmpolk@cisco.com

Subha Dhesikan
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134 USA

mailto: sdhesika@cisco.com

Appendix A - History

This history of how we got to this phase of the development in this document can be traced to the work and choices articulated in the appendix within

draft-ietf-tsvwg-intserv-multiple-tspec

From there, another team developed

draft-lefaucheur-tsvwg-rsvp-multiple-preemption

Then Lou Berger (yeah, blame him!) came up with the bright idea of

combining the two efforts in such a way that one can take a complete Object or element, and replace the primary instance of that within an RSVP message for whatever reason (perhaps the message would be rejected if that piece was not replaced with more reasonable demands on the network; who knows). Anyway, that's how we got here. That's the story and I'm sticking to it... :-p

Network WG
Internet-Draft
Intended status: Informational
Expires: January 9, 2012

James Polk
Cisco
July 9, 2012

The Problem Statement for the Standard
Configuration of DiffServ Service Classes
draft-polk-tsvwg-diffserv-stds-problem-statement-00.txt

Abstract

This document describes the problem statement on two recently proposed expansions to DiffServ. The first of these expansions proposes updating the informational RFC 4594 document to standards track status, while making the necessary changes to make it current; for example, creating more granular traffic treatments, some with new Per Hop Behaviors (PHB). The second proposal defines 6 new DiffServ Codepoints necessary from these new PHBs in the proposal within the first draft.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 9, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Brief Overview of RFC 4594 and RFC 5127	3
2.1 Brief Overview of RFC 4594	3
2.2 Brief Overview of RFC 5127	4
3. Brief Discussion of the RFC 4594 Update Draft	5
4. Conclusion and What's Next	7
5. Acknowledgements	7
6. IANA Considerations	7
7. Security Considerations	8
8. References	8
8.1 Normative References	8
8.2 Informative References	8
Author's Address	9

1. Introduction

Differentiated Services (DiffServ) [RFC2474] creates an IP header marking or indicator with which intermediate nodes (i.e., routers and switches) can make policy decisions. These 6-bit values are called Differentiated Services Codepoint Point (DSCP) values. DSCP values are used to differentiate packet treatment within an intermediate node, not across a network, as the conditions affecting that marking are different within each node. This is called Per Hop Behavior (PHB). In other words, even though a packet has the same DSCP from source to destination, it can and often does experience different treatment depending on the conditions of the nodes it traverses on its journey.

The DiffServ architecture allows for DSCP values within a packet to be changed, or remarked, any number of times. In other words, a packet can have its DSCP remarked at every layer-3 hop throughout the life of that packet. This practice actually occurs infrequently, but it is allowed.

At issue is a combination of the number of networks or endpoints that are choosing to use DiffServ markings, and the number of administrative domains (called "networks" in this document) a packet traverses with different policies for how packet flows of a similar type (e.g., a voice flow, or an email flow, etc.) are to be marked.

The community presently has RFC 4594 [RFC4594], which is an informational guideline on how networks can or should mark certain packet flows with differing traffic characteristics using DiffServ. There are several reasons why this informational RFC lacks the necessary clarity and strength to reach widespread adoption:

- o confusion between RFC 4594 and RFC 5127 [RFC5127], the latter of which is for aggregating many 6-bit DSCP values into a 3-bit (8

value) field used specifically by service provider (SP) networks.

- o some believe both RFCs are for SPs, while others ignore RFC 5127 and use RFC 4594 as if it were standards track or BCP.
- o some believe RFC 5127 is for SPs only, and want RFC 4594 to reduce the number of DSCPs within its guidelines to recommend using only 3 or 4 DSCPs. This seems to stem from a manageability and operational perspective.
- o some know RFC 4594 is informational and do not follow its guidelines specifically because it is informational.
- o some use DSCP values that are not defined within RFC 4594, making mapping between different networks using similar or identical application flows difficult.
- o some believe enterprise networks should not use either RFC except at the edge of their networks, where they directly connect to SP networks.
- o some argue that the services classes guidance per class is too broad and are therefore not sure in which service class a particular application is to reside.

This document is not intended to reach RFC status. Rather, it is to stimulate discussion on both RFC 4594 and 5127 to lessen existing confusion within the community. It should be noted that RFC 4594 has an offered update within TSVWG [ID-4594-UPDATE]. This draft has created some heated discussions within that WG before and during the Paris IETF meeting.

First, we'll discuss briefly RFCs 4594 and 5127 in Section 2. Then we will discuss what the update to RFC 4594 proposes differently and what we expect to happen to RFC 5127 in Section 3.

2. Brief Overview of RFC 4594 and RFC 5127

2.1 Brief Overview of RFC 4594

Essentially, RFC 4594 is a guideline for how to choose which DSCP to use based on the traffic characteristics an application flow needs to experience within a network for optimal performance. RFC 4594 specifically points to several existing standards-track DiffServ RFCs to augment the text in each of those RFCs, without violating any of the rules within each of those documents. RFC 4594:

- o painstakingly lays out definitions and guidelines for each service class.
- o clearly indicates each service class's tolerance to delay, jitter

and packet loss.

- o details the conditioning treatments at the Differentiated Services (DS) edge.
- o categorizes traffic characteristics into 12 service classes utilizing one or more DSCPs:

Network Control	Broadcast Video
Telephony	Low-Latency Data
Signaling	OAM
Multimedia Conferencing	High-throughput Data
Realtime Interactive	Standard
Multimedia Streaming	Low-priority Data

2.2 Brief Overview of RFC 5127

At its barest, RFC 5127 recommends that, of the many service classes described within RFC 4594, each having different traffic characteristics, similar service classes be grouped or aggregated into 3, 4, or 5 markings for SP traversal. This limitation of the number of individual service classes is partly to reduce the number of separate distinctions traversing over their network because SPs have difficulty managing what is deemed 'too many' different classes. Another part for this reduction is customer expectations of meeting contractual Service Level Agreements (SLAs).

To this end, and perhaps because of it, MPLS was designed with only 8 values of priority differentiation, i.e., the 3 EXP bits. To be fair, LAN based IEEE has only a 3-bit priority field as well within its specifications, known as the Priority Code Point (PCP), as part of the 802.1Q header spec. IEEE 802.1e, which defines QoS over Wi-Fi, also only defines 8 levels (called User Priority or UP codes).

The result is to have the IETF within RFC 5127 recommend the following (which is Figure 2 within that RFC):

Treatment Aggregate Behavior	Treatment Aggregate Behavior	DSCP
Network Control	CS (RFC 2474)	CS6
Real-Time	EF (RFC 3246)	EF, CS5, AF41, AF42, AF43, CS4, CS3
Assured Elastic	AF (RFC 2597)	CS2, AF31, AF21, AF11
		AF32, AF22, AF12
		AF33, AF23, AF13
Elastic	Default (RFC 2474)	Default, (CS0)
		CS1

Figure 1: Treatment Aggregate Behavior

RFC 5127 goes on to recommend the marking and treatments on either side of the provider edge remain the same. In other words, the DSCP values remain the same and are used to determine which queue to place the packets into within the aggregates, where the packets are treated the same within that tunnel until the egress provider edge.

Many within enterprise networks do not pay attention to what RFC 5127 says because they are sufficiently removed from dealing with the constraints of very few DSCP values or the need to aggregate DSCP values into groups.

3. Brief Discussion of the RFC 4594 Update Draft

The RFC 4594 update draft [ID-4594-UPDATE] proposes to update what has occurred since RFC 4594 was written (i.e., 2006), in which more granular service classes can be differentiated by application requirements. For example, Figure 2 within RFC 4594 identifies "Telephony" as having 'Fixed-size small packets'. That is not true for today's video flow, therefore it needs to be modified. The update draft currently breaks out audio and video separately to reflect this different, as well as the ability to treat each traffic type differently within a network. Another example is gaming and TCP. The two were believed by most, and it is still believed by many that gaming requires a UDP delivery due to the requirements for timely delivery of packets and that retransmissions would cause delays and bad things to happen to gaming applications. This was proved false within [ID-TCMTF], in which the author of that document

had a presentation showing TCP was used and viable.

[RFC5865] created a new Expedited Forwarding (EF) DSCP value called VOICE-ADMIT, the second time an application is identified within the DiffServ realm. The first was the service class Broadcast Video, which is poorly used within RFC 4594 because other types of flows can be 'broadcast' other than video, such as audio. From this, [ID-4594-UPDATE] moved in two directions:

- o it called out two service classes (audio and video), even though audio and video packets are not the only types of packets within each traffic characteristic.
- o it removed "Video" from the Broadcast service class name.

From the resistance to this proposal within [ID-4594-UPDATE], perhaps other service class label names should be used.

The draft also recognizes the differences in video traffic, even though it is always carried over RTP [RFC3550]. Aside from silence suppression, video traffic varies far more than audio traffic. For example, video is

- o far more variable in bandwidth utilization within the same flow.
- o far more variable in packet size.
- o at different business priorities in some networks based on a configuration. For example, desktop video often is of less important than Telepresence video on the same network. Lacking congestion, the two are treated the same. When congestion exists, one is given priority over the other.

Consequently any service class that contains video needs to account for larger packet size variation than audio, which was equally true in 2006, but not contained in RFC 4594.

Further, with the publication of RFC 5865, the concept of 'capacity admitted' traffic flows have been defined within DiffServ, and are being expanded with the proposal within this new draft [ID-NEW-DSCPS]. There are differing opinions as to whether the realtime Treatment Aggregate in Figure 1 above should also contain these capacity admitted flows, or if 'capacity admitted' traffic flows should have their own Treatment Aggregate containing all realtime capacity admitted traffic. Mixing capacity admitted traffic with unbounded realtime traffic seems to be trouble from a predictability point of view within routers believing they individually understand exactly how much traffic will be traversing each interface and at what rate.

All this said, there is a valid argument to constrain or prevent any DSCP value from being assigned to a single application, mostly due

to the limitation of the overall number of DSCP values available for use. [ID-4594-UPDATE] provides at least several applications per service class (or DSCP); a fact many have overlooked to date.

[ID-4594-UPDATE] is not only about or because of realtime traffic. It is also an overall update to the ideas and guidelines within RFC 4594, with the intent to make that document a standards track document for interoperability purposes.

4. Conclusion and What's Next

Without attempting to fundamentally change the guidelines within RFC 5127, this effort should not be as controversial as it has been, if we understand that those networks that need more granular traffic treatments can be configured with more granularity while not violating the needs of other networks that do not wish to be made aware of the increased treatment differences.

Everyone involved in this discussion needs to have a clear understanding of the difference points of view within the RFC 4594 effort (i.e., the RFC and the update draft) as well as within RFC 5127. One focuses on defining each service class and the other focuses on determining which of the existing service classes go into which aggregate, if present.

We hope to form a BoF on this subject that will explicitly *not* form a working group or produce any documents, or even drafts, but will gather the community from several (if not all) areas, and not just within the transport area. That is the purpose of this draft: to stimulate discussion towards the goal of discussion within the community on DiffServ. If the community does not believe a BoF is necessary, the work will proceed, or not, in TSVWG. Knowing how many within the community have attended TSVWG in each meeting for the last 9 or so years, it is felt that a much wider audience is necessary, given how much impact [ID-4594-UPDATE] can potentially have.

5. Acknowledgements

The author would like to thank Gorrry Fairhurst and David Black for their positive discussions towards the formation of a BoF in Vancouver IETF. The author would also like to thank Paul Jones for doing a valuable proof read to catch points I didn't make clear, as well as identify simple nits I should have caught the nth time I reread this.

6. IANA Considerations

There are no IANA considerations as a result of this document.

7. Security Considerations

There are no security considerations within this document because it will not be progressed beyond this individual contributor stage, and all the specifying will be done in other drafts that will wholly contain all the security considerations for this goal/idea.

8. References

8.1 Normative References

There are no normative references within this document.

8.2 Informative References

- [RFC2474] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers ", RFC 2474, December 1998
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4594] J. Babiarz, K. Chan, F Baker, "Configuration Guidelines for Diffserv Service Classes", RFC 4594, August 2006
- [RFC5127] Chan, K., Babiarz, J., and F. Baker, "Aggregation of DiffServ Service Classes", RFC 5127, February 2008.
- [RFC5865] F. Baker, J. Polk, M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, May 2010
- [ID-4594-UPDATE] J. Polk, "Standard Configuration of DiffServ Service Classes", "work in progress", March 2012
- [ID-NEW-DSCPS] J. Polk, "New Differentiated Services Code Point Assignments for Rich Media Traffic", "work in progress", March 2012
- [ID-TCMTF] J. Saldana, D. Wing, J. Fernandez Navajas, Muthu A M. Perumal, J. Ruiz Mas, "Tunneling Compressed Multiplexed Traffic Flows (TCMTF)", "work in progress", March 2012

Authors' Address

James Polk
3913 Treemont Circle
Colleyville, Texas 76034

Phone: +1.817.271.3552
Email: jmpolk@cisco.com

Network WG
Internet-Draft
Intended status: Standards Track (PS)
Obsoletes: RFC 4594
Updates: RFC 5865
Expires: August 25, 2013

James Polk, ed.
Cisco
Feb, 2013

Standard Configuration of DiffServ Service Classes
draft-polk-tsvwg-rfc4594-update-03.txt

Abstract

This document describes service classes configured with DiffServ and identifies how they are used and how to construct them using Differentiated Services Code Points (DSCPs), traffic conditioners, Per-Hop Behaviors (PHBs), and Active Queue Management (AQM) mechanisms. There is no intrinsic requirement that particular DSCPs, traffic conditioners, PHBs, and AQM be used for a certain service class, but for consistent behavior under the same network conditions, configuring networks as described here is appropriate.

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 25, 2013.

Copyright Notice

Copyright (c) 2012 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. Requirements Notation	
1.2. Expected Use in the Network	
1.3. Service Class Definition	
1.4. Key Differentiated Services Concepts	
1.4.1. Queuing	
1.4.1.1. Priority Queuing	
1.4.1.2. Rate Queuing	
1.4.2. Active Queue Management	
1.4.3. Traffic Conditioning	
1.4.4. Differentiated Services Code Point (DSCP)	
1.4.5. Per-Hop Behavior (PHB)	
1.5. Key Service Concepts	
1.5.1. Default Forwarding (DF)	
1.5.2. Assured Forwarding (AF)	
1.5.3. Expedited Forwarding (EF)	1
1.5.4. Class Selector (CS)	1
1.5.5. Admission Control	1
1.6 What Changes are Proposed Here from RFC 4594?.....	1
2. Service Differentiation	1
2.1. Service Classes	1
2.2. Categorization of User Oriented Service Classes	1
2.3. Service Class Characteristics	1
2.4. Service Classes vs. Treatment Aggregate (from RFC 5127)...	2
2.4.1 Examples of Service Classes in Treatment Aggregates...	2
3. Network Control Traffic	2
3.1. Current Practice in the Internet	2
3.2. Network Control Service Class	2
3.3. OAM Service Class	2
4. User Oriented Traffic	3
4.1. Conversational Service Class Group	3
4.1.1 Audio Service Class	3
4.1.2 Video Service Class	3
4.1.3 Hi-Res Service Class	3
4.2. Realtime-Interactive Service Class	3
4.3. Multimedia Conferencing Service Class	3
4.4. Multimedia Streaming Service Class	3
4.5. Broadcast Video Service Class	4
4.6. Low-Latency Data Service Class	4
4.7. Conversational Signaling Service Class	4
4.8. High-Throughput Data Service Class	4
4.9. Standard Service Class	4
4.10. Low-Priority Data	4
5. Additional Information on Service Class Usage	4
5.1. Mapping for NTP	5

5.2. VPN Service Mapping	5
6. Security Considerations	5
7. Contributing Authors	5
8. Acknowledgements	5
9. References	5
9.1. Normative References	5
9.2. Informative References	5
Author's Address	5
Appendix A - Changes	5

1. Introduction

Differentiated Services [RFC2474][RFC2475] provides the ability to mark/label/classify IP packets differently to distinguish how individual packets need to be treated differently through (or throughout) a network on a per hop basis. Local administrators are who configure each router for which Differentiated Services Code Points (DSCP) are to be treated differently, which are to be ignored (i.e., no differentiated treatment), and which DSCPs are to have their packets remarked (to different DSCPs) as they pass through a router. Local administrators are also who assign which applications, or traffic types, should use which DSCPs to receive the treatment the administrators expect within their network.

What most people fail to understand is that DSCPs provide a per hop behavior (PHB) through that router, but not the previous or next router. In this way of understanding PHB markings, one can understand that Differentiated Services (DiffServ) is not a Quality of Service (QoS) mechanism, but rather a Classification of Service (CoS) mechanism.

For instance, there are 64 possible DSCP values, i.e., using 6 bits of the old Type of Service (TOS) byte [RFC0791]. Each can be configured locally to have greater or less treatment relative to any other DSCP with two exceptions*.

- * Expedited Forwarding (EF) [RFC3246] DSCPs have a treatment requirement that any packet marked within an EF class has to be the next packet transmitted out its egress interface. If there are more than one EF marked packet in the queue, obviously the queue sets the order they are transmitted. Further, if there are more than one EF DSCP, local configuration determines if each are treated the same or differently relate to each other EF DSCP. Currently, there are two Expedited Forwarding DSCPs: EF (101110) [RFC3246] and VOICE-ADMIT (101100) [RFC5865].

- * Class Selector 6 (CS6) [RFC2474] is for routing protocol traffic. There are deemed important because if the network does not transmit and receive its routing protocol traffic in a timely manner, the network stops operating properly.

Not all are configured to mean anything other than best effort forwarding by local administrators of a network. Let us say there are 5 DSCPs configured within network A. Network A's administrator chooses and configures which order (obeying the two exceptions noted above) which application packets are treated differently than any other packets within that network (A). The DSCPs are not fixed to a linear order for relative priority on a per hop basis. Further, and this is often the case, there might be packets with the same DSCP arriving at multiple interfaces of a node, each egressing that node out the same interface. At ingress to this node, everything was fine, with no poor behavior or noticeably excessive amount of packets with the same DSCP. However, at the egress interface, there might not be enough capacity to satisfy the load, thus the departing packets transmit at their maximum rate for that DSCP, but have additional latency due to the overload within that one node. This is called fan-in congestion (or problem). By itself, DiffServ will not remedy this problem for the application that is intolerant to added latency because DiffServ only functions within 1 node at a time.

An additional mechanism is needed to ensure each flow or session receives the amount of packets at its destination that the application requires to perform properly; a mechanism such as IntServ, by way of RSVP [RFC2205] or NSIS [RFC4080]. With this added capability to be session aware, something DiffServ is not, the packets transmitted within a single session have a very good probability of arriving in such a way the receiving application can make full use of each. That said, signaling reservations for each session or flow adds complexity, which creates more work for those who maintain and administer such a network. Adding bandwidth and using DiffServ marking is an easier pill to swallow. The deployment of not few, but more and more audio and (particularly bandwidth hogging) video codecs and their respective application rigidity has caused some to conclude that throwing bandwidth at the problem is no longer acceptable.

With this in mind, this document incorporates five of the six new DSCPs from [ID-DSCP] identified as capacity-admitted DSCPs for most of the service classes in this document. As explained in [ID-DSCP], the five new capacity-admitted DSCPs are from Pool 3. [ID-DSCP] goes further to explain that many layer 2 technologies use fewer bits for marking and prioritization. Instead of six bits like DiffServ, they have three bits, which yields a maximum of 8 values, which tend to line up quite well with the TOS field values. Thus, aggregation of DSCPs is typically accomplished by simply ignoring or reducing the number of bits used to the most significant ones available, such as

EF is 101110, at layer 2 this is merely 101;

Broadcast is 011000, at layer 2 this is merely 011.

However, that was not a premise DiffServ was built upon, to merely

reduce the number of bits. In other words, within DiffServ, XXX is not the same as XXX000 (where XXX is the same binary value in both cases).

This document is originally built upon the RFC 4594 effort, while updating some of the usages and expanding the scope for newer applications that are in use today. The idea in RFC 4594 remains true here, to define a set of service classes, each having unique traffic characteristics, and assigning one or more DSCPs to each service class. As much as the focus could be on the DSCP values, it is not. The focus of this document is the unique traffic characteristics of each service class.

There are many services classes defined in this document, not all will be used in each network at any period of time. This consistency packet markings we talk about is for several reasons, including in a network that does not currently implement a certain service class because they do not have that type of traffic in their network, or that the network merely gives that traffic best effort service. Having a solid guideline to know where to progress or reconfigure a network and endpoints to, say from best effort for a particular traffic type, is a very good thing to do more uniformly than not. A fair amount of burden is placed at DS boundaries needing to keep up with which markings turn into which other markings at both ingress and egress to a network. The same holds true for application developers choosing a default DSCP for their application, lacking a guideline means everyone picks for themselves - and usually with a highly inflated sense of self importance for their application or service.

Another point to make is that there are 20+ service classes defined within the IETF, and that is far too many for most service providers to manage effectively. So, they have formed groups around certain aggregation solutions of service classes. One such aggregation group is based on RFC 5127, which defines what it calls a treatment aggregate, which is taking RFC 4594's service classes and placing them each into one of four treatment aggregates for service providers to handle as a group. SG12 within the ITU-T has an alternative that has nine aggregate groups, so there is work to be done to harmonize aggregates of service classes. This discussion is articulated more in section 2.4. At the end of Section 2.4 we have introduced a series of example configurations which provide examples of how only a few service classes - yet still most treatment aggregates - can be configured in example networks.

Does RFC 4594 need updating? That document is an informational guideline on how networks can or should mark certain packet flows with differing traffic characteristics using DiffServ. There are several reasons why this informational RFC lacks the necessary clarity and strength to reach widespread adoption:

- o confusion between RFC 4594 and RFC 5127 [RFC5127], the latter of

which is for aggregating many 6-bit DSCP values into a 3-bit (8 value) field used specifically by service provider (SP) networks.

- o some believe both RFCs are for SPs, while others ignore RFC 5127 and use RFC 4594 as if it were standards track or BCP.
- o some believe RFC 5127 is for SPs only, and want RFC 4594 to reduce the number of DSCPs within its guidelines to recommend using only 3 or 4 DSCPs. This seems to stem from a manageability and operational perspective.
- o some know RFC 4594 is informational and do not follow its guidelines specifically because it is informational.
- o some use DSCP values that are not defined within RFC 4594, making mapping between different networks using similar or identical application flows difficult.
- o some believe enterprise networks should not use either RFC except at the edge of their networks, where they directly connect to SP networks.
- o some argue that the services classes guidance per class is too broad and are therefore not sure in which service class a particular application is to reside.
- o time has shown that video has become a dominant application on the Internet, and many believe it now requires to be treated uniquely in environments that want to. Video also does not always plan nice with audio, so knowing the two use the same transport (RTP) [RFC3550], a means of separation is in order.

Service class definitions are based on the different traffic characteristics and required performance of the applications/services. There are a greater number of service classes in this document than there were when RFC 4594 [RFC4594] was published (the RFC this document intends to obsolete). The required performance of applications/services has also changed since the publication of RFC 4594, specifically in the area of conversational real time communications. As a result, this document has a greater number of real time applications with more granular set of DSCPs due to their different required performances. Like RFC 4594 before, this approach allows those applications with similar traffic characteristics and performance requirements to be placed in the same service class.

The notion of traffic characteristics and required performance is a per application concept, therefore the label name of each service class remains the same on an end-to-end basis, even if we understand that DiffServ is only a PHB and cannot guarantee anything, even packet delivery at the intended destination node. That said, several applications can be configured to have the same DSCP, or

each have different DSCPs that have the same treatment per hop within a network.

Since RFC 4594 was first published, a new concept has been introduced that will appear throughout this document, including DSCP assignments -- the idea of "admitted" traffic, initially introduced into DiffServ within RFC 5865 [RFC5865]. The VOICE-ADMIT Expedited Forwarding class differentiates itself from the EF Expedited Forwarding by having the packets marked be for admitted traffic. This concept of "admitted" traffic is spread throughout the real time traffic classes.

Thus, the document flow is as follows:

- o maintain the general format of RFC 4594;
- o augment the content with the concept of capacity-admission;
- o incorporate more video into this document, as it has become a dominant application in enterprises and other managed networks, as well as on the open public Internet;
- o reduce the discussion on voice and its examples;
- o articulate the subtle differences learned since RFC 4594 was published.

The goal here is to provide a standard configuration for DiffServ DSCP assignments and expected PHBs for enterprises and other managed networks, as well as towards the public Internet with specific traffic characteristics per Service class/DSCP, and example applications shown for each.

This document describes service classes configured with DiffServ and defines how they can be used and how to construct them using Differentiated Services Code Points (DSCPs), and recommends how to construct them using traffic conditioners, Per-Hop Behaviors (PHBs), and Active Queue Management (AQM) mechanisms. There is no intrinsic requirement that particular traffic conditioners, PHBs, and AQM be used for a certain service class, but as a policy and for interoperability it is useful to apply them consistently.

We differentiate services and their characteristics in Section 2. Network control traffic, as well as user oriented traffic are discussed in Sections 3 and 4, respectively. We analyze the security considerations in Section 6. Section 7 offers a tribute to the authors of RFC 4594, from which this document is based. It is in its own section, and not part of the normal acknowledgements portion of each IETF document.

1.1. Requirements Notation

The key words "SHOULD", "SHOULD NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

1.2. Expected Use in the Network

In the Internet today, corporate LANs and ISP WANs are increasingly utilized, to the point in which network congestion is affecting performance of applications. For this reason, congestion, loss, and variation in delay within corporate LANs and ISP backbones is becoming known to the users collectively as "the network is slow for this application" or just "right now" or "for today". Users do not directly detect network congestion. They react to applications that run slow, or to downloads that take too long in their mind(s). The explosion of video traffic on the internet recently has cause much of this, and is often the application the user is using when they have this slowness.

In the past, application slowness occurred for three very good reasons.

- o the networks the user oriented traffic traverses moves through cycles of bandwidth boom and bandwidth bust, the latter of which become apparent with the periodic deployment of new bandwidth-hungry applications.
- o In access networks, the state is often different. This may be because throughput rates are artificially limited or over-subscribed, or because of access network design trade-offs.
- o Other characteristics, such as database design on web servers (that may create contention points, e.g., in filestore) and configuration of firewalls and routers, often look externally like a bandwidth limitation.

The intent of this document is to provide a standardized marking, plus a conditioning and packet treatment strategy so that it can be configured and put into service on any link that is itself congested.

1.3. Service Class Definition

A "service class" represents a similar set of traffic characteristics for delay, loss, and jitter as packets traverse routers in a network. For example, "High-Throughput Data" service class for store-and-forward applications, or a "Broadcast" service

class for minimally time-shifted IPTV or Internet radio broadcasts. Such a service class may be defined locally in a Differentiated Services (DS) domain, or across multiple DS domains, possibly extending end to end. A goal of this document is to have most/all networks assign the same type of traffic the same for consistency.

A service class is a naming convention which is defined as a word, phrase or initialism/acronym representing a set of necessary traffic characteristics of a certain type of data flow. The necessary characteristics of these traffic flows can be realized by the use of defined per-hop behavior that started with [RFC2474]. The actual specification of the expected treatment of a traffic aggregate within a domain may also be defined as a per-domain behavior (PDB) [RFC3086].

Each domain will locally choose to

- o implement one or more service classes with traffic characteristics as defined here, or
- o implement one or more service classes with similar traffic characteristics as defined here, or
- o implement one or more service classes with similar traffic characteristics as defined here and to aggregate one or more service classes to reduce the number of unique DSCPs within their network, or
- o implement one or more non-standard service classes with traffic characteristics not as defined here, or
- o not use DiffServ within their domain.

For example, low delay, low loss, and minimal jitter may be realized using the EF PHB, or with an over-provisioned AF PHB. This must be done with care as it may disrupt the end-to-end performance required by the applications/services. If the packet sizes are similar within an application, but different between two applications, say small voice packets and large video packets, these two applications may not realize optimum results if merged into the same aggregate if there are any bottlenecks in the network. We provide for this flexibility on a per hop or per domain basis within this document.

This document provides standardized markings for traffic with similar characteristics, and usage expectations for PHBs for specific service classes for their consistent implementation.

The Default Forwarding "Standard" service class is REQUIRED; all other service classes are OPTIONAL. That said, each service class lists traffic characteristics that are expected when using that type of traffic. It is RECOMMENDED that applications and protocols that fit a certain traffic characteristic use the appropriate service

class mark, i.e., the DSCP, for consistent behavior. It is expected that network administrators will base their endpoint application and router configuration choices on the level of service differentiation they require to meet the needs of their customers (i.e., their end-users).

1.4. Key Differentiated Services Concepts

In order to fully understand this document, a reader needs to familiarize themselves with the principles of the Differentiated Services Architecture [RFC2474]. We summarize some key concepts here only to provide convenience for the reader, the referenced RFCs providing the authoritative definitions.

1.4.1. Queuing

A queue is a data structure that holds packets that are awaiting transmission. A router interface can only transmit one packet at a time, however fast the interface speed is. If there is only 1 queue at an interface, the packets are transmitted in the order they are received into that queue - called FIFO, or "first in, first out". Sometimes there is a lag in the time between a packets arrives in the queue and when it is transmitted. This delay might be due to lack of bandwidth, or if there are multiple queues on that interface, because a packet is low in priority relative to other packets that are awaiting to transmit. The scheduler is the system entity that chooses which packet is next in line for transmission when more than one packet are awaiting transmission out the same router interface.

1.4.1.1 Priority Queuing

A priority queuing system is a combination of a set of queues and a scheduler that empties the queues (of packets) in priority sequence. When asked for a packet, the scheduler inspects the highest priority queue and, if there is data present, returns a packet from that queue. Failing that, it inspects the next highest priority queue, and so on. A freeway onramp with a stoplight for one lane that allows vehicles in the high-occupancy-vehicle lane to pass is an example of a priority queuing system; the high-occupancy-vehicle lane represents the "queue" having priority.

In a priority queuing system, a packet in the highest priority queue will experience a readily calculated delay. This is proportional to the amount of data remaining to be serialized when the packet arrived plus the volume of the data already queued ahead of it in the same queue. The technical reason for using a priority queue relates exactly to this fact: it limits delay and variations in delay and should be used for traffic that has that requirement.

A priority queue or queuing system needs to avoid starvation of lower-priority queues. This may be achieved through a variety of means, such as admission control, rate control, or network engineering.

1.4.1.2. Rate Queuing

Similarly, a rate-based queuing system is a combination of a set of queues and a scheduler that empties each at a specified rate. An example of a rate-based queuing system is a road intersection with a stoplight. The stoplight acts as a scheduler, giving each lane a certain opportunity to pass traffic through the intersection.

In a rate-based queuing system, such as Weighted Fair Queuing (WFQ) or Weighted Round Robin (WRR), the delay that a packet in any given queue will experience depends on the parameters and occupancy of its queue and the parameters and occupancy of the queues it is competing with. A queue whose traffic arrival rate is much less than the rate at which it lets traffic depart will tend to be empty, and packets in it will experience nominal delays. A queue whose traffic arrival rate approximates or exceeds its departure rate will tend not to be empty, and packets in it will experience greater delay. Such a scheduler can impose a minimum rate, a maximum rate, or both, on any queue it touches.

1.4.2 Active Queue Management

Active Queue Management, or AQM, is a generic name for any of a variety of procedures that use packet dropping or marking to manage the depth of a queue. The canonical example of such a procedure is Random Early Detection (RED), in that a queue is assigned a minimum and maximum threshold, and the queuing algorithm maintains a moving average of the queue depth. While the mean queue depth exceeds the maximum threshold, all arriving traffic is dropped. While the mean queue depth exceeds the minimum threshold but not the maximum threshold, a randomly selected subset of arriving traffic is marked or dropped. This marking or dropping of traffic is intended to communicate with the sending system, causing its congestion avoidance algorithms to kick in. As a result of this behavior, it is reasonable to expect that TCP's cyclic behavior is desynchronized and that the mean queue depth (and therefore delay) should normally approximate the minimum threshold.

A variation of the algorithm is applied in Assured Forwarding PHB [RFC2597], in that the behavior aggregate consists of traffic with multiple DSCP marks, which are intermingled in a common queue. Different minima and maxima are configured for the several DSCPs separately, such that traffic that exceeds a stated rate at ingress is more likely to be dropped or marked than traffic that is within its contracted rate.

1.4.3 Traffic Conditioning

In addition, at the first router in a network that a packet crosses, arriving traffic may be measured and dropped or marked according to a policy, or perhaps shaped on network ingress, as in "A Rate Adaptive Shaper for Differentiated Services" [RFC2963]. This may be used to bias feedback loops, as is done in "Assured Forwarding PHB" [RFC2597], or to limit the amount of traffic in a system, as is done in "Expedited Forwarding PHB" [RFC3246]. Such measurement procedures are collectively referred to as "traffic conditioners". Traffic conditioners are normally built using token bucket meters, for example with a committed rate and burst size, as in Section 1.5.3 of the DiffServ Model [RFC3290]. The Assured Forwarding PHB [RFC2597] uses a variation on a meter with multiple rate and burst size measurements to test and identify multiple levels of conformance.

Multiple rates and burst sizes can be realized using multiple levels of token buckets or more complex token buckets; these are implementation details. The following are some traffic conditioners that may be used in deployment of differentiated services:

- o For Class Selector (CS) PHBs, a single token bucket meter to provide a rate plus burst size control.
- o For Expedited Forwarding (EF) PHB, a single token bucket meter to provide a rate plus burst size control.
- o For Assured Forwarding (AF) PHBs, usually two token bucket meters configured to provide behavior as outlined in "Two Rate Three Color Marker (trTCM)" [RFC2698] or "Single Rate Three Color Marker (srTCM)" [RFC2697]. The two-rate, three-color marker is used to enforce two rates, whereas the single-rate, three-color marker is used to enforce a committed rate with two burst lengths.

1.4.4 Differentiated Services Code Point (DSCP)

The DSCP is a number in the range 0..63 that is placed into an IP packet to mark it according to the class of traffic it belongs in. These are divided into 3 groups, or pools, defined in RFC 2474, arranged as follows:

- o Pool-1 has 32 values designated for standards assignment (of the form 'xxxxx0').
- o Pool-2 has 16 values designated for experimental or local use only (EXP/LU) assignment (of the form 'xxxx11').
- o Pool-3 has 16 values designated for experimental or local use (EXP/LU) assignment (of the form 'xxxx01').

However, pool-3 is allowed to be assigned for one of two reasons,

#1 - if the values in pool-1 are exhausted, or

#2 - if there is a justifiable reason for assigning a pool-3 DSCP prior to pool-1's exhaustion.

1.4.5 Per-Hop Behavior (PHB)

In the end, the mechanisms described above are combined to form a specified set of characteristics for handling different kinds of traffic, depending on the needs of the application. This document seeks to identify useful traffic aggregates and to specify what PHB should be applied to them.

1.5 Key Service Concepts

While Differentiated Services is a general architecture that may be used to implement a variety of services, three fundamental forwarding behaviors have been defined and characterized for general use. These are basic Default Forwarding (DF) behavior for elastic traffic, the Assured Forwarding (AF) behavior, and the Expedited Forwarding (EF) behavior for real-time (inelastic) traffic. The facts that four code points are recommended for AF and that one code point is recommended for EF are arbitrary choices, and the architecture allows any reasonable number of AF and EF classes simultaneously. The choice of four AF classes and one EF class in the current document is also arbitrary, and operators MAY choose to operate more or fewer of either.

The terms "elastic" and "real-time" are defined in [RFC1633], Section 3.1, as a way of understanding broad-brush application requirements. This document should be reviewed to obtain a broad understanding of the issues in quality of service, just as [RFC2475] should be reviewed to understand the data plane architecture used in today's Internet.

1.5.1 Default Forwarding (DF)

The basic forwarding behaviors applied to any class of traffic are those described in [RFC2474] and [RFC2309]. Best-effort service may be summarized as "I will accept your packets" and is typically configured with some bandwidth guarantee. Packets in transit may be lost, reordered, duplicated, or delayed at random. Generally, networks are engineered to limit this behavior, but changing traffic loads can push any network into such a state.

Application traffic in the internet that uses default forwarding is expected to be "elastic" in nature. By this, we mean that the sender of traffic will adjust its transmission rate in response to

changes in available rate, loss, or delay.

For the basic best-effort service, a single DSCP value is provided to identify the traffic, a queue to store it, and active queue management to protect the network from it and to limit delays.

1.5.2 Assured Forwarding (AF)

The Assured Forwarding PHB [RFC2597] behavior is explicitly modeled on Frame Relay's Discard Eligible (DE) flag or ATM's Cell Loss Priority (CLP) capability. It is intended for networks that offer average-rate Service Level Agreements (SLAs) (as FR and ATM networks do). This is an enhanced best-effort service; traffic is expected to be "elastic" in nature. The receiver will detect loss or variation in delay in the network and provide feedback such that the sender adjusts its transmission rate to approximate available capacity.

For such behaviors, multiple DSCP values are provided (two or three, perhaps more using local values) to identify the traffic, a common queue to store the aggregate, and active queue management to protect the network from it and to limit delays. Traffic is metered as it enters the network, and traffic is variously marked depending on the arrival rate of the aggregate. The premise is that it is normal for users occasionally to use more capacity than their contract stipulates, perhaps up to some bound. However, if traffic should be marked or lost to manage the queue, this excess traffic will be marked or lost first.

1.5.3. Expedited Forwarding (EF)

The intent of Expedited Forwarding PHB [RFC3246] is to provide a building block for low-loss, low-delay, and low-jitter services. It can be used to build an enhanced best-effort service: traffic remains subject to loss due to line errors and reordering during routing changes. However, using queuing techniques, the probability of delay or variation in delay is minimized. For this reason, it is generally used to carry voice and for transport of data information that requires "wire like" behavior through the IP network. Voice is an inelastic "real-time" application that sends packets at the rate the codec produces them, regardless of availability of capacity. As such, this service has the potential to disrupt or congest a network if not controlled. It also has the potential for abuse.

To protect the network, at minimum one SHOULD police traffic at various points to ensure that the design of a queue is not overrun, and then the traffic SHOULD be given a low-delay queue (often using priority, although it is asserted that a rate-based queue can do this) to ensure that variation in delay is not an issue, to meet application needs.

1.5.4 Class Selector (CS)

Class Selector, those DSCPs that end in zeros (xxx000), provide support for historical codepoint definitions and PHB requirement. The CS fields provide a limited backward compatibility with legacy practice, as described in [RFC2474], Section 4. Backward compatibility is addressed in two ways,

- First, there are per-hop behaviors that are already in widespread use (e.g., those satisfying the IPv4 Precedence queuing requirements specified in [RFC1812]), and
- this document will continue to permit their use in DS-compliant networks.

In addition, there are some DSCPs that correspond to historical use of the IP Precedence field,

- CS0 (000000) will remain 'Default Forwarding' (also known as 'Best Effort')
- 11xxxx will remain for routing traffic

and will map to PHBs that meet the general requirements specified in [RFC2474], Section 4.2.2.2.

No attempt is made to maintain backward compatibility with the "DTR" or Type of Service (TOS) bits of the IPv4 TOS octet, as defined in [RFC0791] and [RFC1349].

A DS-compliant network can be deployed exclusively by using one or more CS-compliant PHB groups. Thus, for example, codepoint '011000' would map to the same PHB as codepoint '011010'.

1.5.5 Admission Control

Admission control (including refusal when policy thresholds are crossed) can ensure high-quality communication by ensuring the availability of bandwidth to carry a load. Inelastic real-time flows such as Voice over Internet Protocol (VoIP) (audio) or video conferencing services can benefit from use of an admission control mechanism, as generally the audio or video service is configured with over-subscription, meaning that some users may not be able to make a call during peak periods.

For VoIP (audio) service, a common approach is to use signaling protocols such as SIP, H.323, H.248, MEGACO, along with Resource Reservation Protocol (RSVP) to negotiate admittance and use of network transport capabilities. When a user has been authorized to send voice traffic, this admission procedure has verified that data rates will be within the capacity of the network that it will use.

Many RTP voice and video payloads are inelastic and cannot react to loss or delay in any substantive way. For these payload types, the network needs to police at ingress to ensure that the voice traffic stays within its negotiated bounds. Having thus assured a predictable input rate, the network may use a priority queue to ensure nominal delay and variation in delay.

1.5.5.1 Capacity Admitted (*-Admit)

This is a newer group of traffic types that started with RFC 5865 and the Voice-Admit service type. Voice-Admit is an EF class marking but has capacity-admission always applied to it to ensure each of these flows are managed through a network, though not necessarily on an end-to-end basis. This depends on how many networks each flow transits and the load on each transited network. There are a series of new DSCPs proposed in [ID-DSCP], each specifying unique characteristics necessitating a separate marking from what existing before that document.

This document will import in four new '*-Admit' DSCPs from [ID-DSCP], 2 others that are new but not capacity-admitted, one from RFC 5865, and change the existing usage of 2 DSCPs from RFC 4594. This is discussed throughout the rest of this document.

1.6 What Changes are Proposed Here from RFC 4594?

Changing an entire network DiffServ configuration has proven to be a painful experience for both individuals and companies. It is not done very often, and for good reason. This effort is based on experience learned since the publication of RFC 4594 (circa 2006). Audio, once thought to be ok grouped with video, needs to be in separate service classes. Collaboration has taken off, mostly because of mobility, but also because of a worldwide recession that has limited physical travel, and relying on people to do more with their computers. With that in mind, there has been an explosion in application development for the individual (seems everyone has an "app-store"). The following set of bullets has this world - that needs a robust layer 3 - in mind.

- o Scope of document is changed to tighten it up for standards track consideration.
- o This document explicitly states there is a fundamental requirement that a particular DSCP(s) be used for each service class, each with a recommended set of applications to be used by that service class - at least on that individual's externally facing (public) interface.
- o Created the Conversational group of service classes to focus on realtime, mostly bidirectional communications (unless multicast is

used).

- o "Realtime-Interactive"
Moved to (near) realtime TCP-based apps

Why the change? TCP based transports have proven, in certain environments, to be a bidirectional realtime transport, e.g., for multiplayer gaming and virtual desktops applications.

- o "Audio"
Same as Telephony (which is now gone), adds Voice-Admit for capacity-admitted traffic

Why the change? RFC 5865 (Voice-Admit) needed to be added to the Audio service class. Video needed to be separate from audio, hence the name change from Telephony (which includes video) to just audio.

- o "Video"
NEW for video and audio/video conferencing, was in Multimedia-Conferencing service classification

Why the change? Many networks are using the AF4X for video, but others are throwing anything "multimedia" into the same service class (like elastic TCP flows). Video has become so dominant that it should be what mostly goes into one service class.

- o "Hi-Res"
NEW for video and audio/video conferencing

Why the change? This entirely new service class is for local policy based higher end video (think Telepresence). Without congestion, this service class has the same treatment as Video, but if there is any pushback from the network, Hi-Res (note: not married to the name) has a better PHB.

- o "Multimedia-Conferencing"
Now without audio or human video

Why the change? The change is taking bidirectional human audio and video out of this service class. This is all about non-realtime collaboration - even in conjunction with an audio and/or video flow.

- o "Broadcast"
Remains the same, added CS3-Admit for capacity-admitted

Why the change? Removing the "-Video" from the name because there are so many more flows that are Broadcast in realtime than video.

- o "Low-Latency Data"
Remains the same, adds IM & Presence traffic explicitly

Why the change? Merely explicitly stating a place for some

additional traffic types that otherwise could go elsewhere.

- o "Conversational Signaling" (A/V-Sig)
Was 'Signaling'

Why the change? This change is merely a renaming of a service class, and acknowledgement that some of the previous authors inaccurate beliefs that DSCPs were linearly ordered with those values having a higher value definitely getting better treatment than lower values.

2. Service Differentiation

There are practical limits on the level of service differentiation that should be offered in the IP networks. We believe we have defined a practical approach in delivering service differentiation by defining different service classes that networks may choose to support in order to provide the appropriate level of behaviors and performance needed by current and future applications and services. The defined structure for providing services allows several applications having similar traffic characteristics and performance requirements to be grouped into the same service class. This approach provides a lot of flexibility in providing the appropriate level of service differentiation for current and new, yet unknown applications without introducing significant changes to routers or network configurations when a new traffic type is added to the network.

2.1 Service Classes

Traffic flowing in a network can be classified in many different ways. We have chosen to divide it into two groupings, network control and user/subscriber traffic. To provide service differentiation, different service classes are defined in each grouping. The network control traffic group can further be divided into two service classes (see Section 3 for detailed definition of each service class):

- o "Network Control" for routing and network control function.
- o "OAM" (Operations, Administration, and Management) for network configuration and management functions.

The user/subscriber traffic group is broken down into ten service classes to provide service differentiation for all the different types of applications/services (see Section 4 for detailed definition of each service class):

- o Conversational service group consists of three service classes:
 - Audio, which includes both 'admitted' and 'unadmitted' audio

service classes, is for non-one way (i.e., generally bidirectional) audio media packets between human users of smaller size and at a constant delivery rate.

- Hi-Res Video, which includes both 'admitted' and 'unadmitted' Hi-Res Video service classes, is for video traffic from higher end endpoints between human users necessitating different treatment than from desktop or video phone endpoints. This has a clearly business differentiation, and not a technical differentiation - as both Hi-Res-Video and Video will be treated similarly on the wire when no congestion occurs.
- Video, which includes both 'admitted' and 'unadmitted' video service classes, is for video traffic from lower end endpoints between human users necessitating different treatment than from higher end (i.e., Telepresence) endpoints. This has a clearly business differentiation, and not a technical differentiation - as both Hi-Res-Video and Video will be treated similarly on the wire when no congestion occurs.
- o Conversational Signaling service class is for peer-to-peer and client-server signaling and control functions using protocols such as SIP, H.323, H.248, and Media Gateway Control Protocol (MGCP). This traffic needs to not be starved on the network.

Editor's note: RFC 4594 had this DSCP marking as CS5, but with clearly different characteristics (i.e., no sensitivity to jitter or (unreasonable) delay), this DSCP has been moved to a more appropriate (new) value, defined in [ID-DSCP].

- o Real-Time Interactive, which includes both 'admitted' and 'unadmitted' Realtime-Interactive service class, is for bidirectional variable rate inelastic applications that require low jitter and loss and very low delay, such as interactive gaming applications that use RTP/UDP streams for game control commands, and Virtualized Desktop applications between the user and content source, typically in a centralized data center.
- o Multimedia Conferencing, which includes both 'admitted' and 'unadmitted' multimedia conferencing service class, is for applications that require minimal delay, but not like those of realtime application requirements. This service class can be bursty in nature, as well as not transmit packets for some time. Applications such as presentation data or collaborative application sharing will use this service class.
- o Multimedia Streaming, which includes both 'admitted' and 'unadmitted' multimedia streaming service class, is for one-way bufferable streaming media applications such as Video on Demand (VOD) and webcasts.

- o Broadcast, which includes both 'admitted' and 'unadmitted' broadcast service class, is for inelastic streaming media applications that may be of constant or variable rate, requiring low jitter and very low packet loss, such as broadcast TV and live events, video surveillance, and security.
- o Low-Latency Data service class is for data processing applications such as client/server interactions or Instant Messaging (IM) and Presence data.
- o Conversational Signaling (A/V-Sig) service class is for all signaling messages, whether in-band (i.e., along the data path) or out-of-band (separate from the data path), for the purposes of setting up, maintaining, managing and terminating bi- or multi-directional realtime sessions.
- o High-Throughput Data service class is for store and forward applications such as FTP and billing record transfer.
- o Standard service class, commonly called best effort (BE), is for traffic that has not been identified as requiring differentiated treatment.
- o Low-Priority Data service class, which some could call the scavenger class, is for packet flows where bandwidth assurance is not required.

2.2 Categorization of User Oriented Service Classes

The ten defined user/subscriber service classes listed above can be grouped into a small number of application categories. For some application categories, it was felt that more than one service class was needed to provide service differentiation within that category due to the different traffic characteristic of the applications, control function, and the required flow behavior. Figure 1 provides a summary of service class grouping into four application categories.

Application Control Category

- o The Conversational Signaling service class is intended to be used to control applications or user endpoints. Examples of protocols that would use this service class are SIP, XMPP or H.323 for voice and/or video over IP services. User signaling flows have similar performance requirements as Low-Latency Data, they require a separate DSCP to be distinguished other traffic and allow for a treatment that is unique.

Media-Oriented Category

Due to the vast number of new (in process of being deployed) and already-in-use media-oriented services in IP networks, seven service

classes have been defined.

- o Audio service class is intended for Voice-over-IP (VoIP) services. It may also be used for other applications that meet the defined traffic characteristics and performance requirements.
- o Video service class is intended for Video over IP services. It may also be used for other applications that meet the defined traffic characteristics and performance requirements.
- o Hi-Res service class is intended for higher end video services that have the same traffic characteristics as the video service class, but have a business requirement(s) to be treated differently. One example of this is Telepresence video applications.
- o Realtime-Interactive service class is intended for inelastic applications such as desktop virtualization applications and for interactive gaming.
- o Multimedia Conferencing service class is for everything about or within video conferencing solutions that does not include the voice or (human) video components. Several examples are
 - the presentation data part of an IP conference (call).
 - the application sharing part of an IP conference (call).
 - the whiteboarding aspect of an IP conference (call).

Each of the above can be part of a lower end web-conferencing application or part of a higher end Telepresence video conference. Each also has the ability to reduce their transmission rate on detection of congestion. These flows can therefore be classified as rate adaptive and most often more elastic than their voice and video counterparts.

- o Broadcast Video service class is to be used for inelastic traffic flows specifically with minimal buffering expected by the source or destination, which are intended for broadcast HDTV service, as well as for transport of live video (sports or concerts) and audio events.
- o Multimedia Streaming service class is to be used for elastic multimedia traffic flows where buffering is expected. This is the fundamental difference between the Broadcast and multimedia streaming service classes. Multimedia streaming content is typically stored before being transmitted. It is also buffered at the receiving end before being played out. The buffering is sufficiently large to accommodate any variation in transmission rate that is encountered in the network. Multimedia entertainment over IP delivery services that are being developed

can generate both elastic and inelastic traffic flows; therefore, two service classes are defined to address this space, respectively: Multimedia Streaming and Broadcast Video.

Data Category

The data category is divided into three service classes.

- o Low-Latency Data for applications/services that require low delay or latency for bursty but short-lived flows.
- o High-Throughput Data for applications/services that require good throughput for long-lived bursty flows. High Throughput and Multimedia Streaming are close in their traffic flow characteristics with High Throughput being a bit more bursty and not as long-lived as Multimedia Streaming.
- o Low-Priority Data for applications or services that can tolerate short or long interruptions of packet flows. The Low-Priority Data service class can be viewed as "don't care" to some degree.

Best-Effort Category

- o All traffic that is not differentiated in the network falls into this category and is mapped into the Standard service class. If a packet is marked with a DSCP value that is not supported in the network, it SHOULD be forwarded using the Standard service class.

Figure 1, below, provides a grouping of the defined user/subscriber service classes into four categories, with indications of which ones use an independent flow for signaling or control; type of flow behavior (elastic, rate adaptive, or inelastic); and the last column provides end user Class of Service (CoS) rating as defined in ITU-T Recommendation G.1010.

Application Categories	Service Class	Signaled	Flow Behavior	G.1010 Rating
Application Control	A/V Sig	Not applicable	Inelastic	Responsive
Media-	Realtime Interactive	Yes	Inelastic	Interactive
	Audio	Yes	Inelastic	Interactive
	Video	Yes	Inelastic	Interactive
	Hi-Res	Yes	Inelastic	Interactive
	Multimedia	Yes	Rate	Moderately

Oriented	Conferencing		Adaptive	Interactive
	Broadcast	Yes	Inelastic	Responsive
	Multimedia Streaming	Yes	Elastic	Timely
Data	Low-Latency Data	No	Elastic	Responsive
	Conversational Signaling	No	Elastic or Inelastic	Timely
	High-Throughput Data	No	Elastic	Timely
	Low-Priority Data	No	Elastic	Non-critical
Best Effort	Standard	Not Specified		Non-critical

Figure 1. User/Subscriber Service Classes Grouping

Here is a short explanation of the end user CoS category as defined in ITU-T Recommendation G.1010. User oriented traffic is divided into four different categories, namely, interactive, responsive, timely, and non-critical. An example of interactive traffic is between two humans and is most sensitive to delay, loss, and jitter. Another example of interactive traffic is between two servers where very low delay and loss are needed. Responsive traffic is typically between a human and a server but can also be between two servers. Responsive traffic is less affected by jitter and can tolerate longer delays than interactive traffic. Timely traffic is either between servers or servers and humans and the delay tolerance is significantly longer than responsive traffic. Non-critical traffic is normally between servers/machines where delivery may be delay for period of time.

2.3. Service Class Characteristics

This document specifies what network administrators are to expect when configuring service classes identified by their differing characteristics. Figure 2 identifies these service classes along with their characteristics, as well as the tolerance to loss, delay and jitter for each service class. Properly engineered networks to these PHBs will achieve expected results. That said, not all of the identified service classes are expected in each operator's network.

Service Class Name	Traffic Characteristics	Tolerance to		
		Loss	Delay	Jitter
Network Control	Variable size packets, mostly inelastic short messages, but traffic can also burst (BGP)	Low	Low	Yes
Realtime Interactive	Inelastic, mostly variable rate	Low	Very Low	Low
Audio	Fixed-size small packets, inelastic	Very Low	Very Low	Very Low
Video	Fixed-size small-large packets, inelastic	Very Low	Very Low	Very Low
Hi-Res A/V	Fixed-size small-large packets, inelastic	Very Low	Very Low	Very Low
Multimedia Conferencing	Variable size packets, constant transmit interval, rate adaptive, reacts to loss	Low - Medium	Low - Medium	Low - Medium
Multimedia Streaming	Variable size packets, elastic with variable rate	Low - Medium	Medium	High
Broadcast	Constant and variable rate, inelastic, non-bursty flows	Very Low	Medium	Low
Low-Latency Data	Variable rate, bursty short-lived elastic flows	Low	Low - Medium	Yes
Conversational Signaling	Variable size packets, some what bursty short-lived flows	Low	Low	Yes
OAM	Variable size packets, elastic & inelastic flows	Low	Medium	Yes
High-Throughput Data	Variable rate, bursty long-lived elastic flows	Low	Medium - High	Yes
Standard	A bit of everything	Not Specified		
Low-Priority Data	Non-real-time and elastic	High	High	Yes

Figure 2. Service Class Characteristics

Notes for Figure 2: A "Yes" in the jitter-tolerant column implies that received data is buffered at the endpoint and that a moderate level of server or network-induced variation in delay is not expected to affect the application. Applications that use TCP or SCTP as a transport are generally good examples. Routing protocols and peer-to-peer signaling also fall in this class; although loss can create problems in setting up calls, a moderate level of jitter merely makes call placement a little less predictable in duration.

Service classes indicate the required traffic forwarding treatment in order to meet user, application, and/or network expectations. Section 3 defines the service classes that MAY be used for forwarding network control traffic, and Section 4 defines the service classes that MAY be used for forwarding user oriented traffic with examples of intended application types mapped into each service class. Note that the application types are only examples and are not meant to be all-inclusive or prescriptive. Also, note that the service class naming or ordering does not imply any priority ordering. They are simply reference names that are used in this document with associated QoS behaviors that are optimized for the particular application types they support. Network administrators MAY choose to assign different service class names to the service classes that they will support. Figure 3 defines the RECOMMENDED relationship between service classes and DS codepoint assignment with application examples. It is RECOMMENDED that this relationship be preserved end to end.

Service Class Name	DSCP Name	DSCP Value	Application Examples
Network Control	CS6&CS7	11xxxx	Network routing
Realtime Interactive	CS5, CS5-Admit	101000, 101001	Remote/Virtual Desktop and Interactive gaming
Audio	EF Voice-Admit	101110 101100	Voice bearer
Hi-Res A/V	CS4, CS4-Admit	100000, 100001	Conversational Hi-Res Audio/Video bearer
Video	AF41,AF42 AF43	100010,100100 100110	Audio/Video conferencing bearer
Multimedia Conferencing	MC, MC-Admit	011101, 100101	Presentation Data and App Sharing/Whiteboarding
Multimedia Streaming	AF31,AF32 AF33	011010,011100 011110	Streaming video and audio on demand

Broadcast	CS3, CS3-Admit	011000, 011001	Broadcast TV, live events & video surveillance
Low-Latency Data	AF21,AF22 AF23	010010,010100 010110	Client/server trans., Web- based ordering, IM/Pres
Conversational Signaling	A/V-Sig	010001	Conversational signaling
OAM	CS2	010000	OAM&P
High-Throughput Data	AF11,AF12 AF13	001010,001100 001110	Store and forward applications
Low-Priority Data	CS1	001000	Any flow that has no BW assurance
Best Effort	CS0	000000	Undifferentiated applications

Figure 3. DSCP to Service Class Mapping

Notes for Figure 3:

- o Default Forwarding (DF) and Class Selector 0 (CS0) (i.e., Best Effort) provide equivalent behavior and use the same DS codepoint, '000000'.
- o RFC 2474 identifies any DSCP with a value of 11xxxx to be for network control. This remains true, while it removes 12 DSCPs from the overall pool of 64 available DSCP values (the 4 that are x11 from this group are within pool 2 of RFC 2474, and remain as only experimentally assignable/useable).
- o All PHB names that say "-Admit" are to be used only when a capacity-admission protocol is utilized for that or each traffic flow.

Changes from table 3 of RFC 4594 are as follows:

- o The old term "Signaling" was using CS5 (101000), now is exclusively for the "Conversational Signaling" service group using the DSCP name of "A/V-Sig" (010001), which is newly defined in [ID-DSCP]. This is because CS5 aggregates into the 101xxx aggregate when using layer 2 technologies such as 802.3 Ethernet, 802.11 Wireless Ethernet MPLS, etc - each of which only have 3 bits to mark with. A traffic type that can have very large packets and is not delay sensitive (within reason) is not appropriate for have a 101xxx marking. A REQUIRED behavior for this PHB is that it not be starved in any node.

- o "Conversational" is a new term to include all interactive audio and video. The Conversational service group consists of the audio service class, the video service class and the new Hi-Res service class.
- o "Audio" obsoletes the term "Telephony", which has generally not retained the "video" aspect within the IETF, where video is still commonly called out as a separate thing. Audio retains the nonadmitted traffic PHB of EF (101110), while capacity-admitted audio has been added via the RFC 5865 defined PHB Voice-Admit.
- o "Video" now is AF4x, with AF41 specifically for capacity-admitted video traffic, while AF42 and AF43 are nonadmitted video traffic.
- o "Hi-Res A/V", part of the Conversational service group, is created by [ID-DSCP] for an additional business differentiation interactive video marking for higher end traffic. It is within the 100xxx as CS4 (for nonadmitted traffic) and CS4-Admit (100001) (for capacity-admitted traffic).
- o "Realtime Interactive" is now using CS5 (for nonadmitted traffic), but adds a capacity-admitted DSCP CS5-Admit (101001).
- o "Multimedia Conferencing" is no longer using the AF4x DSCPs, rather it will use the new PHB MC (100101) (for capacity-admitted) and MC-Admit (011101) (for nonadmitted traffic).
- o "Multimedia Streaming" retains using AF3x, however, AF31 is now used for capacity-admitted traffic, while AF32/33 are nonadmitted.
- o "Broadcast" replaces "Broadcast Video" using CS3 (for nonadmitted traffic), and adds a capacity-admitted PHB CS3-Admit (011001).

It is expected that network administrators will base their choice of the service classes that they will support on their need.

Figure 4 provides a summary of DiffServ CoS mechanisms that MUST be used for the defined service classes that are further detailed in Sections 3 and 4 of this document. According to what applications/services need to be differentiated, network administrators MAY choose the service class(es) that need to be supported in their network.

Service Class	DSCP	Conditioning at DS Edge	PHB Used	Queuing	AQM
Network Control	CS6/CS7	See Section 3.1	RFC2474	Rate	Yes
Realtime	CS5,	Police using sr+bs	RFC2474	Rate	No

Interactive	CS5- Admit*		[[ID-DSCP]]		
Audio	EF, Voice- Admit*	Police using sr+bs	RFC3246 RFC5865	Priority	No
Hi-Res A/V	CS4, CS4- Admit*	Police using sr+bs	RFC2474 [[ID-DSCP]]	Priority	No
Video	AF41*, AF42 AF43	Using two-rate, three-color marker (such as RFC 2698)	RFC2597	Rate	Yes per DSCP
Multimedia Conferencing	MC, MC- Admit*	Police using sr+bs	[[ID-DSCP]] [[ID-DSCP]]	Rate	No
Multimedia Streaming	AF31*, AF32 AF33	Using two-rate, three-color marker (such as RFC 2698)	RFC2597	Rate	Yes per DSCP
Broadcast	CS3, CS3- Admit*	Police using sr+bs	RFC2474 [[ID-DSCP]]	Rate	No
Low- Latency Data	AF21 AF22 AF23	Using single-rate, three-color marker (such as RFC 2697)	RFC2597	Rate	Yes per DSCP
Conversational Signaling	AV-Sig	Police using sr+bs	[[ID-DSCP]]	Rate	No
OAM	CS2	Police using sr+bs	RFC2474	Rate	Yes
High- Throughput Data	AF11 AF12 AF13	Using two-rate, three-color marker (such as RFC 2698)	RFC2597	Rate	Yes per DSCP
Standard	DF	Not applicable	RFC2474	Rate	Yes
Low-Priority Data	CS1	Not applicable	RFC3662	Rate	Yes

Figure 4. Summary of CoS Mechanisms Used for Each Service Class

* denotes each DSCP identified for capacity-admission traffic only.

Notes for Figure 4:

- o Conditioning at DS edge means that traffic conditioning is performed at the edge of the DiffServ network where untrusted user devices are connected to two different administrative DiffServ networks.
- o "sr+bs" represents a policing mechanism that provides single rate with burst size control.
- o The single-rate, three-color marker (srTCM) behavior SHOULD be equivalent to RFC 2697, and the two-rate, three-color marker (trTCM) behavior SHOULD be equivalent to RFC 2698.
- o The PHB for Realtime-Interactive service class SHOULD be configured to provide high bandwidth assurance. It MAY be configured as another EF PHB (one capacity-admitted and one non-capacity-admitted, if both are to be used) that uses relaxed performance parameters and a rate scheduler.
- o The PHB for Multimedia Conferencing service class SHOULD be configured to provide high bandwidth assurance. It MAY be configured as another EF PHB (one capacity-admitted and one non-capacity-admitted, if both are to be used) that uses relaxed performance parameters and a rate scheduler.
- o The PHB for Broadcast service class SHOULD be configured to provide high bandwidth assurance. It MAY be configured as another EF PHB (one capacity-admitted and one non-capacity-admitted, if both are to be used) that uses relaxed performance parameters and a rate scheduler.

2.4. Service Classes vs. Treatment Aggregates (from RFC 5127)

There are misconceptions about the differences between RFC 4594 specified service classes, and RFC 5127 specified treatment aggregates. Often the two are conflated, and more often the phrase service class is used to mean both definitions. Almost all of the text previous to this section is used in defining service classes, and how one service class is different than another service class (based on traffic characteristics of the applications). Treatment aggregates are groupings of service classes with similar, but not identical, traffic characteristics to give similar treatment from a SP's network.

Below is taken from appendix of RFC 5127 as its recommended groupings of service classes into aggregates based in RFC 4594 specified traffic characteristic expectations.

+-----+-----+-----+-----+			
Treatment	Treatment	DSCP	
Aggregate	Aggregate		
	Behavior		

Network Control	CS (RFC 2474)	CS6
Real-Time*	EF (RFC 3246)	EF, CS5, AF41, AF42, AF43, CS4, CS3
Assured Elastic	AF (RFC 2597)	CS2, AF31, AF21, AF11 AF32, AF22, AF12 AF33, AF23, AF13
Elastic	Default (RFC 2474)	Default, (CS0) CS1

Figure 5: RFC 5127 Defined Treatment Aggregate Behavior**

*NOTE: The RFC 5865 created VOICE-ADMIT is absence from the above figure because VOICE-ADMIT was created far later than this recommendation was. VOICE-ADMIT is appropriate for the Realtime Traffic Aggregate.

**NOTE: Figure 5 is directly from the appendix of RFC 5127 as that RFC's recommendation for configuration. This draft does not directly affect RFC 5127. That is left for an update to RFC 5127 itself. Based on the WG's take on this draft, RFC 5127 will necessitate an update to match this document's new service classes and additional DSCPs. The number of treatment aggregates are not expected to change in the RFC 5127 Update draft though, with the possible exception of a new treatment aggregate for capacity admitted flows; meaning there *might* be a 5th treatment aggregate proposed.

Treatment Aggregates are designed to nicely fit into technologies that do not have many different treatment levels to use. Here are 3 examples of technologies limited to an 8-value field,

- MPLS with its 3 Traffic Class (TC) bits [RFC5462].
- IEEE LANs with its 8-value Priority Code Point (PCP) field, as part of the 802.1Q header spec [IEEE1Q].
- IEEE 802.1e, which defines QoS over Wi-Fi, also only defines 8 levels (called User Priority or UP codes) [IEEE1E].

Treatment Aggregates are dependent on service classes to exist. Therefore many service classes can exist without the need to consider the use of treatment aggregates or their 8-value technologies. For example, a Layer 3 VPN can be all that is needed

to transit traffic flows, regardless of desired treatment, between enterprise LAN campuses. From this reality, the number of treatment aggregates has no direct bearing on the number of service classes.

2.4.1 Examples of Service Classes in Treatment Aggregates

It is **not** expected that all traffic characteristics are to be experienced across an SP's network for any given customer. For example, if VOICE-ADMIT is added to the Realtime Treatment Aggregate in Figure 5, there are 8 different service classes within the Realtime Treatment Aggregate. It is not expected that all 8 service classes will be deployed by customer networks traversing SP networks. RFC 5127's Treatment Aggregates are a table to configure which service class goes into which treatment aggregate. If there are 8 services classes in the Realtime treatment aggregate, there is very little difference than if there were one service within that same Realtime treatment aggregate - it would still be necessary to configure that treatment aggregate. Thus, it becomes a question of not

"how many service classes are there that go into treatment aggregates?"

but

"how many treatment aggregates have one or more services classes requiring configuration"?

Of the 4 treatment aggregates shown in Figure 5, if there are existing service classes in only 3 of the aggregates, then only 3 treatment aggregates are necessary. Of the 3 following examples, notice that examples 2 and 3 have the same number of treatment aggregates, but example 3 has more applications in their own service classes.

Examples 2 and 3 are made under the following assumptions:

- this draft's Service Classes and DSCP assignments are utilized.
- the new AF-Sig DSCP in the Assured Elastic treatment aggregate.
- the Audio, Video service classes are in the EF treatment aggregate.
- the VOICE-ADMIT DSCP is in the EF treatment aggregate.

2.4.1.1 Example 1 - Simple Voice Configuration/SLA

For example 1, we have an SP running MPLS and has an SLA to deliver Network Control, Voice and everything else is Best Effort. The

following table would apply to this configuration/SLA:

Applications	Service Class	DSCP(s)	Treatment Aggregate
Network Control	Network Control	CS6	Network Control
Voice	Audio	EF	Realtime
Everything else	DF	Default (CS0)	Elastic

Figure 6. Example 1 Configuration

Insert different treatments for this example
(i.e., AQM, RED, WFQ, colors, etc from above charts)

2.4.1.2 Example 2 - Voice/Video/Surveillance Configuration/SLA

For example 1, we have an SP running MPLS and has an SLA to deliver Control, audio, video, surveillance, audio & video signaling, and everything else is BE

Applications	Service Class	DSCP(s)	Treatment Aggregate
Network Control	Network Control	CS6	Network Control
Voice, video, surveillance	Audio, Video, Broadcast	EF, AF42, CS3	Realtime
audio & video signaling	Conversational Signaling	AV-Sig	Assured Elastic
Everything else	DF	Default (CS0)	Elastic

Figure 7. Example 2 Configuration

Insert different treatments for this example
(i.e., AQM, RED, WFQ, colors, etc from above charts)

2.4.1.2 Example 3 - Complex CAC realtime/Surveillance/+apps Configuration/SLA

For example 1, we have an SP running MPLS and has an SLA to deliver

Control, voice, CAC voice, CAC video, streaming, signaling, LL data, Network Mgmt., and everything else is BE (including non-CAC video because it is not authorized or authenticated on network)

Applications	Service Class	DSCP(s)	Treatment Aggregate
Network Control	Network Control	CS6	Network Control
Voice, CAC-Voice CAC-video, surveillance	Audio, Video, Broadcast	Voice-Admit EF, AF41 CS3	Realtime
audio & video signaling, VOD (streaming), Network Mgmt.	Conversational Signaling, Low- Latency Data, Multimedia Streaming, OAM	AV-Sig AF21 AF31 CS2	Assured Elastic
Everything else	DF	Default (CS0)	Elastic

Figure 8. Example 3 Configuration

Insert different treatments for this example
(i.e., AQM, RED, WFQ, colors, etc from above charts)

3. Network Control Traffic

Network control traffic is defined as packet flows that are essential for stable operation of an administered network, as well as the information exchanged between neighboring networks across a peering point where SLAs are in place. Network control traffic is different from user application control (signaling) that may be generated by some applications or services. Network control traffic is mostly between routers and network nodes (e.g., routing or mgmt protocols) that are used for operating, administering, controlling, or managing whole networks, network parts or just network segments. Network Control Traffic may be split into two service classes, i.e., Network Control and OAM.

3.1. Current Practice in the Internet

Based on today's routing protocols and network control procedures that are used in the Internet, we have determined that CS6 DSCP value SHOULD be used for routing and control and that CS7 DSCP value SHOULD be reserved for future use, specifically if needed for future

routing or control protocols. Network administrators MAY use a Local/Experimental DSCP, any value that contains 11xx11; therefore, they may use a locally defined service class within their network to further differentiate their routing and control traffic.

RECOMMENDED Network Edge Conditioning for CS7 DSCP marked packets:

- o Drop or remark 111xxx packets at ingress to DiffServ network domain.
- o 111xxx marked packets SHOULD NOT be sent across peering points. Exchange of control information across peering points SHOULD be done using CS6 DSCP and the Network Control service class.
- o any internally defined 11xxx1 values, valid within that network domain, be remarked to CS6 upon egress at network peering points.

3.2. Network Control Service Class

The Network Control service class is used for transmitting packets between network devices (routers) that require control (routing) information to be exchanged between similar devices within the administrative domain, as well as across a peering point between adjacent administrative domains. Traffic transmitted in this service class is very important as it keeps the network operational, and it needs to be forwarded in a timely manner.

The Network Control service class SHOULD be configured using the DiffServ CS6 PHB, defined in [RFC2474]. This service class MUST be configured so that the traffic receives a minimum bandwidth guarantee, to ensure that the packets always receive timely service. The configured forwarding resources for Network Control service class MUST be such that the probability of packet drop under peak load is very low. The Network Control service class SHOULD be configured to use a Rate Queuing system such as defined in Section 1.4.1.2 of this document.

The following are examples of protocols and applications that MUST use the Network Control service class if present in a network:

- o Routing packet flows: OSPF, BGP, ISIS, RIP.
- o Control information exchange within and between different administrative domains across a peering point where SLAs are in place.
- o LSP setup using CR-LDP and RSVP-TE.

The following protocols and applications MUST NOT use the Network Control service class:

- o User oriented traffic is not allowed to use this service class.

By user oriented traffic, we mean packet flows that originate from user-controlled end points that are connected to the network.

- o even if originating from a server or a device acting on behalf of a user or endpoint,
- o even if it is application or in-band signaling to establish a connection wholly within a single network or across peering points of/to adjacent networks (e.g., creating a tunnel such as a VPN, or data path control signaling).

The following are traffic characteristics of packet flows in the Network Control service class:

- o Mostly messages sent between routers and network servers.
- o Variable size packets, normally one packet at a time, but traffic can also burst (BGP, OSPF, etc).
- o IGMP, hen is used only for the normal multicast routing purpose.

The REQUIRED DSCP marking is CS6 (Class Selector 6).

RECOMMENDED Network Edge Conditioning:

- o At peering points (between two DiffServ networks) where SLAs are in place, CS6 marked packets MUST be policed, e.g., using a single rate with burst size (sr+bs) token bucket policer to keep the CS6 marked packet flows to within the traffic rate specified in the SLA.
- o CS6 marked packet flows from untrusted sources (for example, end user devices) MUST be dropped or remarked at ingress to the DiffServ network. What a network admin remarks this user oriented traffic to is a matter of local policy, and inspection of the packets can determine which application is used for proper marking to a more appropriate DSCP, such as from table 3. of this document.
- o Packets from users/subscribers are not permitted access to the Network Control service classes.

The fundamental service offered to the Network Control service class is enhanced best-effort service with high bandwidth assurance. Since this service class is used to forward both elastic and inelastic flows, the service SHOULD be engineered so that the Active Queue Management (AQM) [RFC2309] is applied to CS6 marked packets.

If RED [RFC2309] is used as an AQM algorithm, the min-threshold specifies a target queue depth, and the max-threshold specifies the queue depth above which all traffic is dropped or ECN marked. Thus,

in this service class, the following inequality should hold in queue configurations:

- o min-threshold CS6 < max-threshold CS6
- o max-threshold CS6 <= memory assigned to the queue

Note: Many other AQM algorithms exist and are used; they should be configured to achieve a similar result.

3.3. OAM Service Class

The OAM (Operations, Administration, and Management) service class is RECOMMENDED for OAM&P (Operations, Administration, and Management and Provisioning) using protocols such as Simple Network Management Protocol (SNMP), Trivial File Transfer Protocol (TFTP), FTP, Telnet, and Common Open Policy Service (COPS). Applications using this service class require a low packet loss but are relatively not sensitive to delay. This service class is configured to provide good packet delivery for intermittent flows.

The OAM service class SHOULD use the Class Selector (CS) PHB defined in [RFC2474]. This service class SHOULD be configured to provide a minimum bandwidth assurance for CS2 marked packets to ensure that they get forwarded. The OAM service class SHOULD be configured to use a Rate Queuing system such as defined in Section 1.4.1.2 of this document.

The following applications SHOULD use the OAM service class:

- o Provisioning and configuration of network elements.
- o Performance monitoring of network elements.
- o Any network operational alarms.

The following are traffic characteristics:

- o Variable size packets.
- o Intermittent traffic flows.
- o Traffic may burst at times.
- o Both elastic and inelastic flows.
- o Traffic not sensitive to delays.

RECOMMENDED DSCP marking:

- o All flows in this service class are marked with CS2 (Class Selector 2).

Applications or IP end points SHOULD pre-mark their packets with CS2 DSCP value. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475].

RECOMMENDED conditioning performed at DiffServ network edge:

- o Packet flow marking (DSCP setting) from untrusted sources (end user devices) SHOULD be verified at ingress to DiffServ network using Multifield (MF) Classification methods, defined in [RFC2475].
- o Packet flows from untrusted sources (end user devices) SHOULD be policed at ingress to DiffServ network, e.g., using single rate with burst size token bucket policer to ensure that the traffic stays within its negotiated or engineered bounds.
- o Packet flows from trusted sources (routers inside administered network) MAY not require policing.
- o Normally OAM&P CS2 marked packet flows are not allowed to flow across peering points. If that is the case, then CS2 marked packets SHOULD be policed (dropped) at both egress and ingress peering interfaces.

The fundamental service offered to "OAM" traffic is enhanced best-effort service with controlled rate. The service SHOULD be engineered so that CS2 marked packet flows have sufficient bandwidth in the network to provide high assurance of delivery. Since this service class is used to forward both elastic and inelastic flows, the service SHOULD be engineered so that Active Queue Management [RFC2309] is applied to CS2 marked packets.

If RED [RFC2309] is used as an AQM algorithm, the min-threshold specifies a target queue depth for each DSCP, and the max-threshold specifies the queue depth above which all traffic with such a DSCP is dropped or ECN marked. Thus, in this service class, the following inequality should hold in queue configurations:

- o min-threshold CS2 < max-threshold CS2
- o max-threshold CS2 <= memory assigned to the queue

Note: Many other AQM algorithms exist and are used; they should be configured to achieve a similar result.

4. User Oriented Traffic

User oriented traffic is defined as packet flows between different users or subscribers, or from servers/nodes on behalf of a user. It is the traffic that is sent to or from end-terminals and that

supports a very wide variety of applications and services, to include traffic about a user or application that assists a user communicate. User oriented traffic can be classified in many different ways. What we have articulated throughout this document is a series of non-exhaustive list of categories for classifying user oriented traffic. We differentiated user oriented traffic that is real-time versus non-real-time, elastic or rate-adaptive versus inelastic, sensitive versus insensitive to loss as well as considering whether the traffic is interactive vs. one way communication, its responsiveness, whether it requires timely delivery, and critical versus non-critical. In the final analysis, we used all of the above for service differentiation, mapping application types that seemed to have different sets of performance sensitivities, and requirements to different service classes.

Network administrators can categorize their applications according to the type of behavior that they require and MAY choose to support all or a subset of the defined service classes. At the same time, we include a public facing default DSCP value, with its associated PHB, that is expected for each traffic type to ensure common or pervasive performance. Figure 3 provides some common applications and the forwarding service classes that best support them, based on their performance requirements.

4.1. Conversational Service Class Group

The Conversational Service Class Group consists of 3 different service classes, audio, video, and Hi-Res. We are describing the media sample, or bearer, packets for applications (e.g., RTP from [RFC3550]) that require bi-directional real-time, very low delay, very low jitter, and very low packet loss for relatively constant-rate traffic sources (inelastic traffic sources). It is RECOMMENDED that RTCP feedback use the same service class and be marked with the same DSCP as the bearer traffic for that (audio and/or video) call. This ensures comparable treatment within the network between endpoints.

The signaling to set-up these bearer flows is part of the Conversational Signaling service group that will be discussed later in Section 4. The following 3 subsections will detail what is expected within each bearer service class.

4.1.1 Audio Service Class

This service class MUST be used for IP Audio service.

The fundamental service offered to traffic in the Audio service class is minimum jitter, delay, and packet loss service up to a specified upper bound. There are two PHBs, both EF based, for the Audio service class:

Nonadmitted Audio traffic - MUST use the EF DSCP [RFC3246], and

is for traffic that has not had any capacity admission signaling performed for that flow or session.

Capacity-Admitted Audio traffic - MUST use the Voice-Admit DSCP [RFC5865], and is for traffic that has had any capacity admission signaling performed for that flow or session, e.g., RSVP [RFC2205] or NSIS [RFC4080].

The capacity-admitted Audio traffic operation is similar to an ATM CBR service, which has guaranteed bandwidth and which, if it stays within the negotiated rate, experiences nominal delay and no loss.

The nonadmitted Audio traffic, on the other hand, has had no such explicit guarantee, but has a favorable PHB ensuring high probability of delivery as well as nominal delay and no loss - implicitly assuming there is not too much like marked traffic between users within a flow.

There are two typical scenarios in which audio calls are established, on the public open Internet using protocols such as SIP, XMPP or H.323, or in more managed networks like enterprises or certain service providers which offer a audio service with some feature benefits and take part in the call signaling. These SPs or enterprises also use protocols like SIP, XMPP, H.323, but also use H.248/MEGACO and MGCP.

On the open Internet, typically there is no SP actively involved in the session set-up of calls, and therefore no servers providing assistance or features to help one user contact another user. Often, this traffic is marked or remarked with the DF (i.e., Best Effort) DSCP.

In more managed networks in which one of more operators have active servers aiding the audio call set-up, where DiffServ can be used and preserved to differentiate traffic, networks are offering a service, therefore need to do some, or a lot of engineering to ensure that capacity offered to one or more applications does not exceed the load to the network. Otherwise, the operator will have unhappy users, at least for that application's usage. This is true for any application, but is especially true for inelastic applications in which the application is rigid in its delivery requirements. Audio bearer traffic is typically such an application, video is another such application, but we will get to video in the next subsection.

When a user in a managed network has been authorized to send Audio traffic (i.e., call initiation via the operator's servers was not rejected), the call admission procedure should have verified that the newly admitted flow will be within the capacity of the Audio service class forwarding capability in the network. Capacity verification is a non-trivial thing, and can either be implicitly assumed by the call server(s) based on the operator's network design, or it can be explicitly signaled from an in-data-path

signaling mechanism that verifies the capacity is available now for this call, for each call made within that network. In the latter case, those that do not have verifiable network capacity along the data path are rejected. An in between means method is for call servers to count calls between two or more endpoints. By topologically understanding where the caller and called party is and have configured a known maximum it will allow between the two locations. This is especially true over WAN links that have far less capacity than LAN links or core parts of a network. Network operators will need to understand the topology between any two callers to ensure the appropriate amount of bandwidth is available for an expected number of simultaneous audio calls.

Once more than one bandwidth amount can be used for audio calls, for example - by allowing more than one codec with different bandwidths per codec for such calls, network engineering becomes more difficult. Since the inelastic nature of RTP payloads from this class do not react well to loss or significant delay in any substantive way, the Audio service class MUST forward packets as soon as possible.

The Audio service class that does not have capacity admission performed in the data path MUST use the Expedited Forwarding (EF) PHB, as defined in [RFC3246], so that all packets are forwarded quickly. The Audio service class that does have capacity admission performed in the data path MUST use the Voice-Admit PHB, as defined in [RFC5865], so that all packets are forwarded quickly. The Audio service class SHOULD be configured to use a Priority Queuing system such as that defined in Section 1.4.1.1 of this document.

The following applications SHOULD use the Audio service class:

- o VoIP (G.711, G.729, iLBC and other audio codecs).
- o Voice-band data over IP (modem, fax).
- o T.38 fax over IP.
- o Circuit emulation over IP, virtual wire, etc.
- o IP Virtual Private Network (VPN) service that specifies single-rate, mean network delay that is slightly longer than network propagation delay, very low jitter, and a very low packet loss.

The following are traffic characteristics:

- o Mostly fixed-size packets for VoIP (30, 60, 70, 120 or 200 bytes in size).
- o Packets emitted at constant time intervals.

- o Admission control of new flows is provided by Audio call server, media gateway, gatekeeper, edge router, end terminal, access node or in-data-path signaling that provides flow admission control function.

Applications or IP end points SHOULD pre-mark their packets with EF or Voice-Admit DSCP value, whichever is appropriate. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475].

The RECOMMENDED DSCP marking is EF for nonadmitted audio flows, and Voice-Admit for capacity-admitted flows for the following applications:

- o VoIP (G.711, G.729 and other codecs).
- o Voice-band data over IP (modem and fax).
- o T.38 fax over IP.
- o Circuit emulation over IP, virtual wire, etc.

RECOMMENDED Network Edge Conditioning:

- o Packet flow marking (DSCP setting) from untrusted sources (end user devices) SHOULD be verified at ingress to DiffServ network using Multifield (MF) Classification methods, defined in [RFC2475]. If untrusted, the network edge SHOULD know if capacity-admission has been applied, since the edge router will have taken part in the admission signaling; therefore will know whether EF or Voice-Admit is the proper marking for that flow.
- o Packet flows from untrusted sources (end user devices) SHOULD be policed at ingress to DiffServ network, e.g., using single rate with burst size token bucket policer to ensure that the Audio traffic stays within its negotiated bounds.
- o Policing is OPTIONAL for packet flows from trusted sources whose behavior is ensured via other means (e.g., administrative controls on those systems).
- o Policing of Audio packet flows across peering points where SLA is in place is OPTIONAL as Audio traffic will be controlled by admission control mechanism between peering points.

The fundamental service offered to "Audio" traffic is enhanced best-effort service with controlled rate, very low delay, and very low loss. The service MUST be engineered so that EF marked packet flows have sufficient bandwidth in the network to provide guaranteed delivery. Otherwise, the service will have in place an explicit capacity-admission signaling protocol such as RSVP or NSIS and thus

mark the packets within the flow as Voice-Admit. Normally traffic in this service class does not respond dynamically to packet loss. As such, Active Queue Management [RFC2309] SHOULD NOT be applied to EF marked packet flows.

4.1.2 Video Service Class

The Video service class is for bidirectional applications that require real-time service for both constant and rate-adaptive traffic. SIP and H.323/V2 (and later) versions of video conferencing equipment with constant and dynamic bandwidth adjustment are such applications. The traffic sources in this service class either have a fixed bandwidth requirement (e.g., MPEG2, etc.), or have the ability to dynamically change their transmission rate (e.g., MPEG4/H.264, etc.) based on feedback from the receiver. This feedback SHOULD be accomplished using RTCP [RFC3550]. One approach for this downspeeding has the receiver detect packet loss, thus signaling in an RTCP message to the source the indication of lost (or delayed or out of order) packets in transit. When necessary the source then selects a lower rate encoding codec. When available, the source merely sends less data, resulting in lower resolution of the same visual display.

The Video service class is not for video downloads, webcasts, or single directional video or audio/video traffic of any kind. It is for human-to-human visual interaction between two users, or more if an MTP is used.

Typical video conferencing configurations negotiate the setup of audio/video session using protocols such as SIP and H.323. Just as with networks that have audio traversing them, video typically traverses the same two types of networks: the open big "I" Internet, in which most every type of traffic is best effort (DF), or on a more managed network such as an enterprise or SP's managed network in which servers within either network take part in the call signaling, thereby offering the video service.

When a user in a managed network has been authorized to send video traffic (i.e., call initiation via the operator's servers was not rejected), the call admission procedure should have verified that the newly admitted flow will be within the capacity of the video service class forwarding capability in the network. Capacity verification is a non-trivial thing, and can either be implicitly assumed by the call server(s) based on the operator's network design, or it can be explicitly signaled from an in-data-path signaling mechanism that verifies the capacity is available now for this call, for each call made within that network. In the latter case, those that do not have verifiable network capacity along the data path are rejected. An in between means method is for call servers to count calls between two or more endpoints. By topologically understanding where the caller and called party is and

have configured a known maximum it will allow between the two locations. Video is larger in bandwidth than audio, and the difference can be significant. For example, for a single G.711 audio call that is 80kbps, an associated video bandwidth for the same call can easily be 4Mbps. This is especially true over WAN links that have far less capacity than LAN links or core parts of a network. Network operators will need to understand the topology between any two callers to ensure the appropriate amount of bandwidth is available for an expected number of simultaneous video and/or audio/video calls.

Note that it is OPTIONALLY the case in these networks that the accompanying audio for the video call will be marked as the video is marked (i.e., using the same DSCP), but not always. One reason this has been done is for lip-sync.

The Video service class MUST use the Assured Forwarding (AF) PHB, defined in [RFC2597]. This service class MUST be configured to provide a bandwidth assurance for AF41, AF42, and AF43 marked packets to ensure that they get forwarded. The Video service class SHOULD be configured to use a Rate Queuing system for AF42 and AF43 traffic flows, such as that defined in Section 1.4.1.2 of this document. However, AF41 MUST be designated as the DSCP for use when capacity-admission signaling has been used, such as RSVP or NSIS, to guarantee delivery through the network. AF42 and AF43 will be used for non-admitted video calls, as well as overflows from AF41 sources that send more packets than they have negotiated bandwidth for that call.

The following applications MUST use the Video service class:

- o SIP and H.323/V2 (and later) versions of video conferencing applications (interactive video).
- o Video conferencing applications with rate control or traffic content importance marking.
- o Interactive, time-critical, and mission-critical applications.

NOTE with regards to the above bullet: this usage SHOULD be minimized, else the video traffic will suffer - unless this is engineered into the topology.

The following are traffic characteristics:

- o Variable size packets (i.e., small to large in size).
- o The higher the resolution or change rate between each image, the higher the duration of large packets.
- o Usually constant inter-packet time interval.

- o Can be Variable rate in transmission.
- o Source is capable of reducing its transmission rate based on being told receiver is detecting packet loss (e.g., via RTCP).

Applications or IP end points SHOULD pre-mark their packets with DSCP values as shown below. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475] and mark all packets as AF4x. Note: In this case, the two-rate, three-color marker will be configured to operate in Color-Blind mode.

Mandatory DSCP marking when performed by router closest to source:

- o AF41 = up to specified rate "A", which is dedicated to non-Hi-Res capacity-admitted video traffic.

Note the audio of an A/V call can be marked AF41 as well.

- o AF42 = all non-Hi-Res video traffic marked AF41 in excess of specified rate "A", or new non-admitted video traffic but below specified rate "B".
- o AF43 = in excess of specified rate "B".
- o Where "A" < "B".

Note: One might expect "A" to approximate the peak rates of sum of all admitted video flows, plus the sum of the mean rates and "B" to approximate the sum of the peak rates of those same two flows.

Mandatory DSCP marking when performed by SIP or H.323/V2 videoconferencing equipment:

- o AF41 = SIP or H.323 video conferencing audio stream RTP.
- o AF41 = SIP or H.323 video conferencing video control RTCP.
- o AF41 = SIP or H.323 video conferencing video stream up to specified rate "A".
- o AF42 = SIP or H.323 video conferencing video stream in excess of specified rate "A" but below specified rate "B".
- o AF42 = SIP or H.323 video conferencing video control RTCP, for those video streams that were generated using AF42.
- o AF43 = SIP or H.323 video conferencing video stream in excess of specified rate "B".

- o AF43 = SIP or H.323 video conferencing video control RTCP, for those video streams that were generated using AF43.
- o Where "A" < "B".

Mandatory conditioning performed at DiffServ network edge:

- o The two-rate, three-color marker SHOULD be configured to provide the behavior as defined in trTCM [RFC2698].
- o If packets are marked by trusted sources or a previously trusted DiffServ domain and the color marking is to be preserved, then the two-rate, three-color marker SHOULD be configured to operate in Color-Aware mode.
- o If the packet marking is not trusted or the color marking is not to be preserved, then the two-rate, three-color marker SHOULD be configured to operate in Color-Blind mode.

The fundamental service offered to nonadmitted "Video" traffic is enhanced best-effort service with controlled rate and delay. The fundamental service offered to capacity-admitted "Video" traffic is a guaranteed service using in-data-path signaling to ensure expected delivery in a timely manner. For a non-admitted video conferencing service, if a 1% packet loss detected at the receiver triggers an encoding rate change, thus dropping to the next lower provisioned video encoding rate then Active Queue Management [RFC2309] SHOULD be used primarily to switch the video encoding rate under congestion, changing from high rate to lower rate, i.e., 1472 kbps to 768 kbps. This rule applies to all AF42 and 43 flows. The probability of loss of AF41 traffic MUST NOT exceed the probability of loss of AF42 traffic, which in turn MUST NOT exceed the probability of loss of AF43 traffic.

Capacity-admitted video service should not result in packet loss. However, administratively this MAY be allowed to cause a purposeful downspeeding event (i.e., a change in resolution or a change in codec) to occur due to congestion.

If RED [RFC2309] is used as an AQM algorithm, the min-threshold specifies a target queue depth for each DSCP, and the max-threshold specifies the queue depth above which all traffic with such a DSCP is dropped or ECN marked. Thus, in this service class, the following inequality should hold in queue configurations:

- o min-threshold AF43 < max-threshold AF43
- o max-threshold AF43 <= min-threshold AF42
- o min-threshold AF42 < max-threshold AF42
- o max-threshold AF42 <= min-threshold AF41

- o min-threshold AF41 < max-threshold AF41
- o max-threshold AF41 <= memory assigned to the queue

Note: This configuration tends to drop AF43 traffic before AF42 and AF42 before AF41. Many other AQM algorithms exist and are used; they should be configured to achieve a similar result.

4.1.3 Hi-Res Service Class

The Hi-Res service class is for higher end (i.e., deemed 'more important') bidirectional applications that require real-time service for both constant and rate-adaptive traffic. There are two PHBs, both EF based, for the Hi-Res video conferencing service class:

Nonadmitted Hi-Res traffic - MUST use the CS4 DSCP [RFC2474], and is for traffic that has not had any capacity admission signaling performed for that flow or session.

Capacity-Admitted Hi-Res traffic - MUST use the CS4-Admit DSCP [ID-DSCP], and is for traffic that has had any capacity admission signaling performed for that flow or session, e.g., RSVP [RFC2205] or NSIS [RFC4080].

The capacity-admitted Hi-Res video conferencing traffic operation is similar to an ATM CBR service, which has guaranteed bandwidth and which, if it stays within the negotiated rate, experiences nominal delay and no loss.

SIP and H.323/V2 (and later) versions of video conferencing equipment with constant and dynamic bandwidth adjustment are such applications. The traffic sources in this service class either have a fixed bandwidth requirement (e.g., MPEG2), or have the ability to dynamically change their transmission rate (e.g., MPEG4/H.264) based on feedback from the receiver. This feedback SHOULD be accomplished using RTCP [RFC3550]. One approach for this downspeeding has the receiver detect packet loss, thus signaling in an RTCP message to the source the indication of lost (or delayed or out of order) packets in transit. When necessary the source then selects a lower rate encoding codec. When available, the source merely sends less data, resulting in lower resolution of the same visual display.

The Hi-Res service class, as with the Video service class, is not for video downloads, webcasts, or single directional video or audio/video traffic of any kind. It is for human-to-human visual interaction between two users, or more if a video conference bridge is used.

Typical Hi-Res video conferencing configurations negotiate the setup

of audio/video session using protocols such as SIP and H.323. Hi-Res video conferencing is generally not over the big "I" Internet, rather nearly exclusively over more managed networks such as an enterprise or special purpose SP's managed network in which servers within either network take part in the call signaling, thereby offering the video service. In addition, typically this type of audio/video service has high business expectations for minimized packet loss, pixilation or other issues with the audio/video experience. In the recent past, entire T3s have been dedicated to a signal Hi-Res call; sometimes one T3 per site of a multi-site video conference.

Hi-Res video conferencing often has larger in bandwidth than the typical video call. The audio portion can be increased as well, as stereo capabilities are often necessary to provide an in-room experience from a distance. The difference can be significant (or another step up from just a typical video service). For example, for a single G.711 audio call that is 80kbps, a Hi-Res conference usually runs G.722 wideband audio at 256kbps. Typical video delivery is up to 4Mbps, whereas a Hi-Res conference can have three 1080p/30fps widescreen displays requiring at least 12Mbps, with a burst capability of much more.

If there were no congestion on the wire, the expected treatment between a video service and a Hi-Res conference would be the same. However, it is typically the case that the Hi-Res conferencing flows have more rigid requirements for quality and business-wise, need to be experience far less errors than the regular video service on the same network.

Note that it is likely the case in these networks that the accompanying audio to the Hi-Res video call will be marked as the Hi-Res video is marked (i.e., using the same DSCP).

The Hi-Res service class MUST use the Class Selector 5 (CS4) PHB, defined in [RFC2474], for non-capacity-admitted conferences. While the capacity-admitted Hi-Res conferences MUST use the CS4-Admit PHB, defined in [ID-DSCP]. This service class MUST be configured to provide a bandwidth assurance for CS4 and CS4-Admit marked packets to ensure that they get forwarded. The Hi-Res service class SHOULD be configured to use a Priority Queuing system such as that defined in Section 1.4.1.1 of this document. Further, CS4-Admit will be designated as the DSCP for use when capacity-admission signaling has been used, such as RSVP or NSIS, to guarantee delivery through the network. CS4 will be used for non-admitted Hi-Res conferences, as well as overflows from CS4-Admit sources that send more packets than they have negotiated bandwidth for that call.

The following applications MUST use the Hi-Res service class:

- o SIP and H.323/V2 (and later) versions of Hi-Res video conferencing applications (interactive Hi-Res video).

- o Video conferencing applications with rate control or traffic content importance marking.

The following are traffic characteristics:

- o Variable size packets.
- o The higher the resolution or change rate between each image, the higher the duration of large packets.
- o Usually constant inter-packet time interval.
- o Can be Variable rate in transmission.
- o Source is capable of reducing its transmission rate based on being told receiver is detecting packet loss.

Applications or IP end points SHOULD pre-mark their packets with DSCP values as shown below. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475] and mark all packets as AF4x.

Mandatory DSCP marking when performed by router closest to source:

- o CS4-Admit = up to specified rate "A", which is dedicated to capacity-admitted Hi-Res traffic.

Note the audio of an A/V call can be marked CS4-Admit as well.

- o CS4 = all video traffic marked CS4-Admit in excess of specified rate "A", or new non-admitted video traffic but below specified rate "B".
- o Where "A" < "B".

Note: One might expect "A" to approximate the peak rates of sum of all admitted video flows, plus the sum of the mean rates and "B" to approximate the sum of the peak rates of those same two flows.

Mandatory DSCP marking when performed by SIP or H.323/V2 videoconferencing equipment:

- o CS4-Admit = SIP or H.323 video conferencing audio stream RTP/UDP.
- o CS4-Admit = SIP or H.323 video conferencing video control RTCP/TCP.
- o CS4-Admit = SIP or H.323 video conferencing video stream up to specified rate "A".

- o CS4 = SIP or H.323 video conferencing video stream in excess of specified rate "A" but below specified rate "B".
- o Where "A" < "B".

Mandatory conditioning performed at DiffServ network edge:

- o The two-rate, three-color marker SHOULD be configured to provide the behavior as defined in trTCM [RFC2698].
- o If packets are marked by trusted sources or a previously trusted DiffServ domain and the color marking is to be preserved, then the two-rate, three-color marker SHOULD be configured to operate in Color-Aware mode.
- o If the packet marking is not trusted or the color marking is not to be preserved, then the two-rate, three-color marker SHOULD be configured to operate in Color-Blind mode.

The fundamental service offered to nonadmitted "Hi-Res" traffic is enhanced best-effort service with controlled rate and delay. The fundamental service offered to capacity-admitted "Hi-Res" traffic is a guaranteed service using in-data-path signaling to ensure expected or timely delivery. Capacity-admitted video service SHOULD NOT result in packet loss. However, administratively this MAY be allowed to cause a purposeful downspeeding event (i.e., a change in resolution or a change in codec) to occur.

4.2. Realtime-Interactive Service Class

The Realtime-Interactive service class is for bidirectional applications that require low loss and jitter and very low delay for constant or variable rate inelastic traffic sources. Interactive gaming applications that do not have the ability to change encoding rates or to mark packets with different importance indications is one good example of such an application. Another set of applications is virtualized desktop applications in which a remote user has a keyboard, mouse and display monitor, but the desktop is virtualized with the memory/processor/applications back in a common data center, requiring near instantaneous feedback on the user's monitor of any changes caused by the application or an action by the user. Rich media protocols for voice and video MUST NOT use the Realtime-Interactive service class, but rather the appropriate service class from the Conversational service group discussed early in Section 4.1.

The Realtime-Interactive service class will use two PHBs:

Nonadmitted Realtime-Interactive traffic - MUST use the CS5 DSCP [RFC2474], and is for traffic that has not had any capacity

admission signaling performed for that flow or session.

Capacity-Admitted Realtime-Interactive traffic - MUST use the CS5-Admit DSCP [ID-DSCP], and is for traffic that has had any capacity admission signaling performed for that flow or session, e.g., RSVP [RFC2205] or NSIS [RFC4080].

The capacity-admitted Realtime-Interactive traffic operation is similar to an ATM CBR service, which has guaranteed bandwidth and which, if it stays within the negotiated rate, experiences nominal delay and no loss.

Either of the above service classes can be configured as EF based by using a relaxed performance parameter and a rate scheduler.

When a user/endpoint has been authorized to start a new session (i.e., joins a networked game or logs onto a virtualized workstation), the admission procedure should have verified that the newly admitted data rates will be within the engineered capacity of the Realtime-Interactive service class. The bandwidth in the core network and the number of simultaneous Realtime-Interactive sessions that can be supported SHOULD be engineered to control traffic load for this service.

This service class SHOULD be configured to provide a high assurance for bandwidth for CS5 PHB, defined in [RFC2474], or CS5-Admit [ID-DSCP] for guaranteed service through a capacity-admission signaling protocol. The Realtime-Interactive service class SHOULD be configured to use a Rate Queuing system such as that defined in Section 1.4.1.2 of this document. Note that either Realtime-Interactive PHB MAY be configured as another EF PHB, specifically CS5-Admit, that uses a relaxed performance parameter and a rate scheduler, in the priority queue as defined in Section 1.4.1.1 of this document.

The following applications MUST use the Realtime-Interactive service class:

- o Interactive gaming and control.
- o Remote Desktop applications
- o Virtualized Desktop applications.
- o Application server-to-application server non-bursty data transfer requiring very low delay.
- o Inelastic, interactive, time-critical, and mission-critical applications requiring very low delay.

The following are traffic characteristics:

- o Variable size packets.
- o Variable rate, though sometimes bursty, which will require engineering of the network to accommodate.
- o Application is sensitive to delay variation between flows and sessions.
- o Lost packets, if any, are usually ignored by application.

RECOMMENDED DSCP marking:

- o All non-admitted flows in this service class are marked with CS5 (Class Selector 5).
- o All capacity-admitted flows in this service class are marked with CS5-Admit.

Applications or IP end points SHOULD pre-mark their packets with CS5 or CS5-Admit DSCP value, depending on whether a capacity-admission signaling protocol is used for a flow. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475].

RECOMMENDED conditioning performed at DiffServ network edge:

- o Packet flow marking (DSCP setting) from untrusted sources (end user devices) SHOULD be verified at ingress to DiffServ network using Multifield (MF) Classification methods defined in [RFC2475].
- o Packet flows from untrusted sources (end user devices) SHOULD be policed at ingress to DiffServ network, e.g., using single rate with burst size token bucket policer to ensure that the traffic stays within its negotiated or engineered bounds.
- o Packet flows from trusted sources (application servers inside administered network) MAY not require policing.
- o Policing of packet flows across peering points MUST adhere to the Service Level Agreement (SLA).

The fundamental service offered to nonadmitted "Realtime-Interactive" traffic is enhanced best-effort service with controlled rate and delay. The fundamental service offered to capacity-admitted "Realtime-Interactive" traffic is a guaranteed service using in-data-path signaling to ensure expected or timely delivery. Capacity-admitted Realtime-Interactive service SHOULD NOT result in packet loss. The service SHOULD be engineered so that CS5 marked packet flows have sufficient bandwidth in the network to provide high assurance of delivery. Normally, traffic in this

service class does not respond dynamically to packet loss. As such, Active Queue Management [RFC2309] SHOULD NOT be applied to CS5 marked packet flows.

4.3. Multimedia Conferencing Service Class

The Multimedia Conferencing service class is for applications that have a low to medium tolerance to delay, and are rate adaptive to lost packets in transit from sources. Presentation Data applications that are operational in conjunction with an audio/video conference is one good example of such an application. Another set of applications is application sharing or whiteboarding applications, also in conjunction to an A/V conference. In either case, the audio & video part of the flow MUST NOT use the Multimedia Conferencing service class, rather the more appropriate service class within the Conversational service group discussed earlier in Section 4.1.

The Multimedia Conferencing service class will use two PHBs:

Nonadmitted Multimedia Conferencing traffic - MUST use the (new) MC DSCP [ID-DSCP], and is for traffic that has not had any capacity admission signaling performed for that flow or session.

Capacity-Admitted Multimedia Conferencing traffic - MUST use the (new) MC-Admit DSCP [ID-DSCP], and is for traffic that has had any capacity admission signaling performed for that flow or session, e.g., RSVP [RFC2205] or NSIS [RFC4080].

The capacity-admitted Multimedia Conferencing traffic operation is similar to an ATM CBR service, which has guaranteed bandwidth and which, if it stays within the negotiated rate, experiences nominal delay and no loss.

When a user/endpoint initiates a presentation data, application sharing or whiteboarding session, it will typically be part of an audio or audio/video conference such as web-conferencing or an existing Telepresence call. The authorization procedure SHOULD be controlled through the coordinated effort to bind the A/V call with the correct Multimedia Conferencing packet flow through some use of identifiers not in scope of this document. The managed network this flow traverse and the number of simultaneous Multimedia Conferencing sessions that can be supported SHOULD be engineered to control traffic load for this service.

The non-capacity admitted Multimedia Conferencing service class SHOULD use the new MC PHB, defined in [ID-DSCP]. This service class SHOULD be configured to provide a high assurance for bandwidth for CS5 marked packets to ensure that they get forwarded. The Multimedia Conferencing service class SHOULD be configured to use a

Rate Queuing system such as that defined in Section 1.4.1.2 of this document. Note that this service class MAY be configured as another EF PHB that uses a relaxed performance parameter, a rate scheduler, and MC-Admit DSCP value, which MUST use the priority queue as defined in Section 1.4.1.1 of this document.

The following applications MUST use the Multimedia Conferencing service class:

- o Presentation Data applications, which can utilize vector graphics, raster graphics or video delivery.
- o Virtualized Desktop applications.
- o Application server-to-application server non-bursty data transfer requiring very low delay.

The following are traffic characteristics:

- o Variable size packets.
- o Variable rate, though sometimes bursty, which will require engineering of the network to accommodate.
- o Application is sensitive to delay variation between flows and sessions.
- o Lost packets, if any, can be ignored by the application.

RECOMMENDED DSCP marking:

- o All non-admitted flows in this service class are marked with the new MC DSCP.
- o All capacity-admitted flows in this service class are marked with MC-Admit.

Applications or IP end points SHOULD pre-mark their packets with the MC DSCP value. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475].

RECOMMENDED conditioning performed at DiffServ network edge:

- o Packet flow marking (DSCP setting) from untrusted sources (end user devices) SHOULD be verified at ingress to DiffServ network using Multifield (MF) Classification methods defined in [RFC2475].
- o Packet flows from untrusted sources (end user devices) SHOULD be policed at ingress to DiffServ network, e.g., using single rate with burst size token bucket policer to ensure that the traffic

stays within its negotiated or engineered bounds.

- o Packet flows from trusted sources (application servers inside administered network) MAY not require policing.
- o Policing of packet flows across peering points MUST adhere to the Service Level Agreement (SLA).

The fundamental service offered to nonadmitted "Multimedia Conferencing" traffic is enhanced best-effort service with controlled rate and delay. The fundamental service offered to capacity-admitted "Multimedia Conferencing" traffic is a guaranteed service using in-data-path signaling to ensure expected or timely delivery. Capacity-admitted Multimedia Conferencing service SHOULD NOT result in packet loss. The service SHOULD be engineered so that Multimedia Conferencing marked packet flows have sufficient bandwidth in the network to provide high assurance of delivery. Normally, traffic in this service class does not respond dynamically to packet loss. As such, Active Queue Management [RFC2309] SHOULD NOT be applied to MC or MC-Admit marked packet flows.

4.4. Multimedia Streaming Service Class

The Multimedia Streaming service class is RECOMMENDED for applications that require near-real-time packet forwarding of variable rate elastic traffic sources that are not as delay sensitive as applications using the Broadcast service class. Such applications include streaming audio and video, some video (movies) on-demand applications, and non-interactive webcasts. In general, the Multimedia Streaming service class assumes that the traffic is buffered at the source/destination; therefore, it is less sensitive to delay and jitter.

The Multimedia Streaming service class MUST use the Assured Forwarding (AF3x) PHB, defined in [RFC2597]. This service class MUST be configured to provide a minimum bandwidth assurance for AF31, AF32, and AF33 marked packets to ensure that they get forwarded. The Multimedia Streaming service class SHOULD be configured to use Rate Queuing system for AF32 and AF33 traffic flows, such as that defined in Section 1.4.1.2 of this document. However, AF31 MUST be designated as the DSCP for use when capacity-admission signaling has been used, such as RSVP or NSIS, to guarantee delivery through the network. AF32 and AF33 will be used for non-admitted streaming flows, as well as overflows from AF31 sources that send more packets than they have negotiated bandwidth for that call.

The following applications SHOULD use the Multimedia Streaming service class:

- o Buffered streaming audio (unicast).

- o Buffered streaming video (unicast).
- o Non-interactive Webcasts.
- o IP VPN service that specifies two rates and is less sensitive to delay and jitter.

The following are traffic characteristics:

- o Variable size packets.
- o The higher the rate, the higher the density of large packets.
- o Variable rate.
- o Elastic flows.
- o Some bursting at start of flow from some applications, as well as an expected stepping up and down on the rate of the flow based on changes in resolution due to network conditions.

Applications or IP end points SHOULD pre-mark their packets with DSCP values as shown below. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475], and mark all packets as AF3x. Note: In this case, the two-rate, three-color marker will be configured to operate in Color-Blind mode.

RECOMMENDED DSCP marking:

- o AF31 = up to specified rate "A".
- o AF32 = all traffic marked AF31 in excess of specified rate "A", or new AF32 traffic but below specified rate "B".
- o AF33 = in excess of specified rate "B".
- o Where "A" < "B".

Note: One might expect "A" to approximate the peak rates of sum of all streaming flows, plus the sum of the mean rates and "B" to approximate the sum of the peak rates of those same two flows.

RECOMMENDED conditioning performed at DiffServ network edge:

- o The two-rate, three-color marker SHOULD be configured to provide the behavior as defined in trTCM [RFC2698].
- o If packets are marked by trusted sources or a previously trusted DiffServ domain and the color marking is to be preserved, then

the two-rate, three-color marker SHOULD be configured to operate in Color-Aware mode.

- o If the packet marking is not trusted or the color marking is not to be preserved, then the two-rate, three-color marker SHOULD be configured to operate in Color-Blind mode.

The fundamental service offered to nonadmitted "Multimedia Streaming" traffic is enhanced best-effort service with controlled rate and delay. The fundamental service offered to capacity-admitted "Multimedia Streaming" traffic is a guaranteed service using in-data-path signaling to ensure expected delivery in a reasonable manner. The service SHOULD be engineered so that AF31 marked packet flows have sufficient bandwidth in the network to provide high assurance of delivery. Since the AF3x traffic is elastic and responds dynamically to packet loss, Active Queue Management [RFC2309] SHOULD be used primarily to reduce forwarding rate to the minimum assured rate at congestion points, unless AF31 has had a capacity-admission signaling protocol applied to the flow, such as RSVP or NSIS.

If a capacity-admission signaling protocol applied to the AF31 flow, which SHOULD be the case always, the AF31 PHB MAY be configured as another EF PHB that uses a relaxed performance parameter and a rate scheduler, in the priority queue as defined in Section 1.4.1.1 of this document.

The probability of loss of AF31 traffic MUST NOT exceed the probability of loss of AF32 traffic, which in turn MUST NOT exceed the probability of loss of AF33.

If RED [RFC2309] is used as an AQM algorithm, the min-threshold specifies a target queue depth for each DSCP, and the max-threshold specifies the queue depth above which all traffic with such a DSCP is dropped or ECN marked. Thus, in this service class, the following inequality MUST hold in queue configurations:

- o min-threshold AF33 < max-threshold AF33
- o max-threshold AF33 <= min-threshold AF32
- o min-threshold AF32 < max-threshold AF32
- o max-threshold AF32 <= min-threshold AF31
- o min-threshold AF31 < max-threshold AF31
- o max-threshold AF31 <= memory assigned to the queue

Note#1: this confirmation MUST be modified if AF31 has a capacity-admission signaling protocol applied to those flows, and the above will only apply to AF32 and AF33, while

AF31 (theoretically) has no packet loss.

Note#2: This configuration tends to drop AF33 traffic before AF32 and AF32 before AF31. Note: Many other AQM algorithms exist and are used; they SHOULD be configured to achieve a similar result.

4.5. Broadcast Service Class

The Broadcast service class is RECOMMENDED for applications that require near-real-time packet forwarding with very low packet loss of constant rate and variable rate inelastic traffic sources that are more delay sensitive than applications using the Multimedia Streaming service class. Such applications include broadcast TV, streaming of live audio and video events, some video-on-demand applications, and video surveillance. In general, the Broadcast service class assumes that the destination end point has a dejitter buffer, for video application usually a 2 - 8 video-frame buffer (66 to several hundred of milliseconds), thus expecting far less buffering before play-out than Multimedia Streaming, which can buffer in the seconds to minutes (to hours).

The Broadcast service class will use two PHBs:

Nonadmitted Broadcast traffic - MUST use the CS3 DSCP [RFC2474], and is for traffic that has not had any capacity admission signaling performed for that flow or session.

Capacity-Admitted Broadcast traffic - MUST use the CS3-Admit DSCP [ID-DSCP], and is for traffic that has had any capacity admission signaling performed for that flow or session, e.g., RSVP [RFC2205] or NSIS [RFC4080].

The capacity-admitted Broadcast traffic operation is similar to an ATM CBR service, which has guaranteed bandwidth and which, if it stays within the negotiated rate, experiences nominal delay and no loss.

Either of the above service classes can be configured as EF based by using a relaxed performance parameter and a rate scheduler.

When a user/endpoint initiates a new Broadcast session (i.e., starts an Internet radio application, starts a live Internet A/V event or a camera comes online to do video-surveillance), the admission procedure should be verified within the application that triggers the flow. The newly admitted data rates will SHOULD be within the engineered capacity of the Broadcast service class within that network. The bandwidth in the core network and the number of simultaneous Broadcast sessions that can be supported SHOULD be engineered to control traffic load for this service.

This service class SHOULD be configured to provide high assurance for bandwidth for CS3 marked packets to ensure that they get forwarded. The Broadcast service class SHOULD be configured to use Rate Queuing system such as that defined in Section 1.4.1.2 of this document. Note that either Broadcast PHB MAY be configured as another EF PHB, specifically CS3-Admit, that uses a relaxed performance parameter and a rate scheduler, in the priority queue as defined in Section 1.4.1.1 of this document.

The following applications SHOULD use the Broadcast service class:

- o Video surveillance and security (unicast).
- o TV broadcast including HDTV (likely multicast, but can be unicast).
- o Video on demand (unicast) with control (virtual DVD).
- o Streaming of live audio events (both unicast and multicast).
- o Streaming of live video events (both unicast and multicast).

The following are traffic characteristics:

- o Variable size packets.
- o The higher the rate, the higher the density of large packets.
- o Mixture of variable rate and constant rate flows.
- o Fixed packet emission time intervals.
- o Inelastic flows.

RECOMMENDED DSCP marking:

- o All non-admitted flows in this service class are marked with CS3 (Class Selector 3).
- o All capacity-admitted flows in this service class are marked with CS3-Admit.
- o In some cases, such as those for security and video surveillance applications, it is NOT RECOMMENDED, but allowed to use a different DSCP marking.

If so, then locally user definable (EXP/LU) codepoints in the range '011x11' MAY be used to provide unique traffic identification. The locally administrator definable (EXP/LU, from pool 2 of RFC 2474) codepoint(s) MAY be associated with the PHB that is used for CS3 or CS3-Admit traffic. Furthermore, depending on the network scenario, additional network edge

conditioning policy MAY be needed for the EXP/LU codepoint(s) used.

Applications or IP end points SHOULD pre-mark their packets with CS3 or CS3-Admit DSCP value. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475].

RECOMMENDED conditioning performed at DiffServ network edge:

- o Packet flow marking (DSCP setting) from untrusted sources (end user devices) SHOULD be verified at ingress to DiffServ network using Multifield (MF) Classification methods defined in [RFC2475].
- o Packet flows from untrusted sources (end user devices) SHOULD be policed at ingress to DiffServ network, e.g., using single rate with burst size token bucket policer to ensure that the traffic stays within its negotiated or engineered bounds.
- o Packet flows from trusted sources (application servers inside administered network) MAY not require policing.
- o Policing of packet flows across peering points MUST be performed to the Service Level Agreement (SLA) of those peering entities.

The fundamental service offered to "Broadcast" traffic is enhanced best-effort service with controlled rate and delay. The fundamental service offered to capacity-admitted "Broadcast" traffic is a guaranteed service using in-data-path signaling to ensure expected or timely delivery. Capacity-admitted Broadcast service SHOULD NOT result in packet loss. The service SHOULD be engineered so that CS3 and CS3-Admit marked packet flows have sufficient bandwidth in the network to provide high assurance of delivery. Normally, traffic in this service class does not respond dynamically to packet loss. As such, Active Queue Management [RFC2309] SHOULD NOT be applied to CS3 marked packet flows.

4.6. Low-Latency Data Service Class

The Low-Latency Data service class is RECOMMENDED for elastic and responsive typically client-/server-based applications. Applications forwarded by this service class are those that require a relatively fast response and typically have asymmetrical bandwidth need, i.e., the client typically sends a short message to the server and the server responds with a much larger data flow back to the client. The most common example of this is when a user clicks a hyperlink (~ few dozen bytes) on a web page, resulting in a new web page to be loaded (Kbytes or MBs of data). This service class is configured to provide good response for TCP [RFC1633] short-lived flows that require real-time packet forwarding of variable rate

traffic sources.

The Low-Latency Data service class SHOULD use the Assured Forwarding (AF) PHB, defined in [RFC2597]. This service class SHOULD be configured to provide a minimum bandwidth assurance for AF21, AF22, and AF23 marked packets to ensure that they get forwarded. The Low-Latency Data service class SHOULD be configured to use a Rate Queuing system such as that defined in Section 1.4.1.2 of this document.

The following applications SHOULD use the Low-Latency Data service class:

- o Client/server applications.
- o Systems Network Architecture (SNA) terminal to host transactions (SNA over IP using Data Link Switching (DLSw)).
- o Web-based transactions (E-commerce).
- o Credit card transactions.
- o Financial wire transfers.
- o Enterprise Resource Planning (ERP) applications (e.g., SAP/BaaN).
- o VPN service that supports Committed Information Rate (CIR) with up to two burst sizes.
- o Instant Messaging and Presence protocols (e.g., SIP, XMPP).

The following are traffic characteristics:

- o Variable size packets.
- o Variable packet emission rate.
- o With packet bursts of TCP window size.
- o Short traffic bursts.
- o Source capable of reducing its transmission rate based on detection of packet loss at the receiver or through explicit congestion notification.

Applications or IP end points SHOULD pre-mark their packets with DSCP values as shown below. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475] and mark all packets as AF2x. Note: In this case, the single-rate, three-color marker will be configured to operate in Color-Blind mode.

RECOMMENDED DSCP marking:

- o AF21 = flow stream with packet burst size up to "A" bytes.
- o AF22 = flow stream with packet burst size in excess of "A" but below "B" bytes.
- o AF23 = flow stream with packet burst size in excess of "B" bytes.
- o Where "A" < "B".

RECOMMENDED conditioning performed at DiffServ network edge:

- o The single-rate, three-color marker SHOULD be configured to provide the behavior as defined in srTCM [RFC2697].
- o If packets are marked by trusted sources or a previously trusted DiffServ domain and the color marking is to be preserved, then the single-rate, three-color marker SHOULD be configured to operate in Color-Aware mode.
- o If the packet marking is not trusted or the color marking is not to be preserved, then the single-rate, three-color marker SHOULD be configured to operate in Color-Blind mode.

The fundamental service offered to "Low-Latency Data" traffic is enhanced best-effort service with controlled rate and delay. The service SHOULD be engineered so that AF21 marked packet flows have sufficient bandwidth in the network to provide high assurance of delivery. Since the AF2x traffic is elastic and responds dynamically to packet loss, Active Queue Management [RFC2309] SHOULD be used primarily to control TCP flow rates at congestion points by dropping packets from TCP flows that have large burst size. The probability of loss of AF21 traffic MUST NOT exceed the probability of loss of AF22 traffic, which in turn MUST NOT exceed the probability of loss of AF23. Explicit Congestion Notification (ECN) [RFC3168] MAY also be used with Active Queue Management.

If RED [RFC2309] is used as an AQM algorithm, the min-threshold specifies a target queue depth for each DSCP, and the max-threshold specifies the queue depth above which all traffic with such a DSCP is dropped or ECN marked. Thus, in this service class, the following inequality should hold in queue configurations:

- o min-threshold AF23 < max-threshold AF23
- o max-threshold AF23 <= min-threshold AF22
- o min-threshold AF22 < max-threshold AF22
- o max-threshold AF22 <= min-threshold AF21

- o min-threshold AF21 < max-threshold AF21
- o max-threshold AF21 <= memory assigned to the queue

Note: This configuration tends to drop AF23 traffic before AF22 and AF22 before AF21. Many other AQM algorithms exist and are used; they should be configured to achieve a similar result.

4.7. Conversational Signaling Service Class

The Signaling service class is MUST be limited to delay-sensitive signaling traffic only, and then only applying to signaling that involves the Conversational service group. Audio signaling includes signaling between IP phone and soft-switch, soft-client and soft-switch, and media gateway and soft-switch as well as peer-to-peer using various protocols. Video and Hi-Res signaling includes video endpoint to video endpoint, as well as to Media transfer Point (MTP), to call control server(S), etc. This service class is intended to be used for control of voice and video sessions and applications. Protocols using this service class require a relatively fast response, as there are typically several messages of different sizes sent for control of the session. This service class is configured to provide good response for short-lived, intermittent flows that require real-time packet forwarding. This is not the service class for Instant Messaging (IM), that's within the bounds of the Low-Latency Data service class. The Conversational Signaling service class MUST be configured so that the probability of packet drop or significant queuing delay under peak load is very low in IP network segments that provide this interface.

The Conversational Signaling service class MUST use the new A/V-Sig PHB, defined in [ID-DSCP]. This service class MUST be configured to provide a minimum bandwidth assurance for A/V-Sig marked packets to ensure that they get forwarded. In other words, this service class MUST NOT be starved from transmission within a reasonable timeframe, given that the entire Conversational service group depends on these signaling messages successful delivery. Network engineering SHOULD be done to ensure there is roughly 1-4% available per node interface that audio and video traverse. Local conditions MUST be considered when determining exactly how much bandwidth is given to this service class. The Conversational Signaling service class SHOULD be configured to use a Rate Queuing system such as that defined in Section 1.4.1.2 of this document.

The following applications SHOULD use the Conversational Signaling service class:

- o Peer-to-peer IP telephony signaling (e.g., SIP, H.323, XMPP).
- o Peer-to-peer signaling for multimedia applications (e.g., SIP, H.323, XMPP).

- o Peer-to-peer real-time control function.
- o Client-server IP telephony signaling using H.248, MEGACO, MGCP, IP encapsulated ISDN, or other proprietary protocols.
- o Signaling to control IPTV applications using protocols such as IGMP.
- o Signaling flows between high-capacity telephony call servers or soft switches using protocol such as SIP-T. Such high-capacity devices may control thousands of telephony (VoIP) calls.
- o Signaling for one-way video flows, such as RTSP [RFC2326].
- o IGMP, when used for multicast session control such as channel changing in IPTV systems.
- o OPTIONALLY, this service class can be used for on-path reservation signaling for the traffic flows that will use the "admitted" DSCPs. The alternative is to have the on-path signaling (for reservations) use the DSCP within that service class. This provides a similar treatment of the signaling to the data flow, which might be desired.

The following are traffic characteristics:

- o Variable size packets, normally one packet at a time.
- o Intermittent traffic flows.
- o Traffic may burst at times.
- o Delay-sensitive control messages sent between two end points.

RECOMMENDED DSCP marking:

- o All flows in this service class are marked with A/V-Sig.

Applications or IP end points SHOULD pre-mark their packets with A/V-Sig DSCP value. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475].

RECOMMENDED conditioning performed at DiffServ network edge:

- o Packet flow marking (DSCP setting) from untrusted sources (end user devices) SHOULD be verified at ingress to DiffServ network using Multifield (MF) Classification methods defined in [RFC2475].

- o Packet flows from untrusted sources (end user devices) SHOULD be policed at ingress to DiffServ network, e.g., using single rate with burst size token bucket policer to ensure that the traffic stays within its negotiated or engineered bounds.
- o Packet flows from trusted sources (application servers inside administered network) MAY not require policing.
- o Policing of packet flows across peering points in which each peer is participating in the call set-up MUST be performed to the Service Level Agreement (SLA).

The fundamental service offered to "Conversational Signaling" traffic is enhanced best-effort service with controlled rate and delay. The service SHOULD be engineered so that A/V-Sig marked packet flows have sufficient bandwidth in the network to provide high assurance of delivery and low delay. Normally, traffic in this service class does not respond dynamically to packet loss. As such, Active Queue Management [RFC2309] SHOULD NOT be applied to A/V-Sig marked packet flows.

4.8. High-Throughput Data Service Class

The High-Throughput Data service class is RECOMMENDED for elastic applications that require timely packet forwarding of variable rate traffic sources and, more specifically, is configured to provide good throughput for TCP longer-lived flows. TCP [RFC1633] or a transport with a consistent Congestion Avoidance Procedure [RFC2581] [RFC3782] normally will drive as high a data rate as it can obtain over a long period of time. The FTP protocol is a common example, although one cannot definitively say that all FTP transfers are moving data in bulk.

The High-Throughput Data service class SHOULD use the Assured Forwarding (AF) PHB, defined in [RFC2597]. This service class SHOULD be configured to provide a minimum bandwidth assurance for AF11, AF12, and AF13 marked packets to ensure that they are forwarded in a timely manner. The High-Throughput Data service class SHOULD be configured to use a Rate Queuing system such as that defined in Section 1.4.1.2 of this document.

The following applications SHOULD use the High-Throughput Data service class:

- o Store and forward applications.
- o File transfer applications (e.g., FTP, HTTP, etc).
- o Email.
- o VPN service that supports two rates (committed information rate

and excess or peak information rate).

The following are traffic characteristics:

- o Variable size packets.
- o Variable packet emission rate.
- o Variable rate.
- o With packet bursts of TCP window size.
- o Source capable of reducing its transmission rate based on detection of packet loss at the receiver or through explicit congestion notification.

Applications or IP end points SHOULD pre-mark their packets with DSCP values as shown below. If the end point is not capable of setting the DSCP value, then the router topologically closest to the end point SHOULD perform Multifield (MF) Classification, as defined in [RFC2475], and mark all packets as AF1x. Note: In this case, the two-rate, three-color marker will be configured to operate in Color-Blind mode.

RECOMMENDED DSCP marking:

- o AF11 = up to specified rate "A".
- o AF12 = in excess of specified rate "A" but below specified rate "B".
- o AF13 = in excess of specified rate "B".
- o Where "A" < "B".

RECOMMENDED conditioning performed at DiffServ network edge:

- o The two-rate, three-color marker SHOULD be configured to provide the behavior as defined in trTCM [RFC2698].
- o If packets are marked by trusted sources or a previously trusted DiffServ domain and the color marking is to be preserved, then the two-rate, three-color marker SHOULD be configured to operate in Color-Aware mode.
- o If the packet marking is not trusted or the color marking is not to be preserved, then the two-rate, three-color marker SHOULD be configured to operate in Color-Blind mode.

The fundamental service offered to "High-Throughput Data" traffic is enhanced best-effort service with a specified minimum rate. The service SHOULD be engineered so that AF11 marked packet flows have

sufficient bandwidth in the network to provide assured delivery. It can be assumed that this class will consume any available bandwidth and that packets traversing congested links may experience higher queuing delays or packet loss. Since the AF_{lx} traffic is elastic and responds dynamically to packet loss, Active Queue Management [RFC2309] SHOULD be used primarily to control TCP flow rates at congestion points by dropping packets from TCP flows that have higher rates first. The probability of loss of AF₁₁ traffic MUST NOT exceed the probability of loss of AF₁₂ traffic, which in turn MUST NOT exceed the probability of loss of AF₁₃. In such a case, if one network customer is driving significant excess and another seeks to use the link, any losses will be experienced by the high-rate user, causing him to reduce his rate. Explicit Congestion Notification (ECN) [RFC3168] MAY also be used with Active Queue Management.

If RED [RFC2309] is used as an AQM algorithm, the min-threshold specifies a target queue depth for each DSCP, and the max-threshold specifies the queue depth above which all traffic with such a DSCP is dropped or ECN marked. Thus, in this service class, the following inequality should hold in queue configurations:

- o min-threshold AF₁₃ < max-threshold AF₁₃
- o max-threshold AF₁₃ <= min-threshold AF₁₂
- o min-threshold AF₁₂ < max-threshold AF₁₂
- o max-threshold AF₁₂ <= min-threshold AF₁₁
- o min-threshold AF₁₁ < max-threshold AF₁₁
- o max-threshold AF₁₁ <= memory assigned to the queue

Note: This configuration tends to drop AF₁₃ traffic before AF₁₂ and AF₁₂ before AF₁₁. Many other AQM algorithms exist and are used; they should be configured to achieve a similar result.

4.9. Standard Service Class

The Standard service class is RECOMMENDED for traffic that has not been classified into one of the other supported forwarding service classes in the DiffServ network domain. This service class provides the Internet's "best-effort" forwarding behavior. This service class typically has minimum bandwidth guarantee.

The Standard service class MUST use the Default Forwarding (DF) PHB, defined in [RFC2474], and SHOULD be configured to receive at least a small percentage of forwarding resources as a guaranteed minimum. This service class SHOULD be configured to use a Rate Queuing system such as that defined in Section 1.4.1.2 of this document.

The following applications SHOULD use the Standard service class:

- o Network services, DNS, DHCP, BootP.
- o Any undifferentiated application/packet flow transported through the DiffServ enabled network.

The following is a traffic characteristic:

- o Non-deterministic, mixture of everything.

The RECOMMENDED DSCP marking is DF (Default Forwarding) '000000'.

Network Edge Conditioning:

There is no requirement that conditioning of packet flows be performed for this service class.

The fundamental service offered to the Standard service class is best-effort service with active queue management to limit overall delay. Typical configurations SHOULD use random packet dropping to implement Active Queue Management [RFC2309] or Explicit Congestion Notification [RFC3168], and MAY impose a minimum or maximum rate on the queue.

If RED [RFC2309] is used as an AQM algorithm, the min-threshold specifies a target queue depth, and the max-threshold specifies the queue depth above which all traffic is dropped or ECN marked. Thus, in this service class, the following inequality should hold in queue configurations:

- o min-threshold DF < max-threshold DF
- o max-threshold DF <= memory assigned to the queue

Note: Many other AQM algorithms exist and are used; they should be configured to achieve a similar result.

4.10. Low-Priority Data

The Low-Priority Data service class serves applications that run over TCP [RFC0793] or a transport with consistent congestion avoidance procedures [RFC2581] [RFC3782] and that the user is willing to accept service without guarantees. This service class is specified in [RFC3662] and [QBSS].

The following applications MAY use the Low-Priority Data service class:

- o Any TCP based-application/packet flow transported through the DiffServ enabled network that does not require any bandwidth assurances.

The following is a traffic characteristic:

- o Non-real-time and elastic.

Network Edge Conditioning:

There is no requirement that conditioning of packet flows be performed for this service class.

The RECOMMENDED DSCP marking is CS1 (Class Selector 1).

The fundamental service offered to the Low-Priority Data service class is best-effort service with zero bandwidth assurance. By placing it into a separate queue or class, it may be treated in a manner consistent with a specific Service Level Agreement.

Typical configurations SHOULD use Explicit Congestion Notification [RFC3168] or random loss to implement Active Queue Management [RFC2309].

If RED [RFC2309] is used as an AQM algorithm, the min-threshold specifies a target queue depth, and the max-threshold specifies the queue depth above which all traffic is dropped or ECN marked. Thus, in this service class, the following inequality should hold in queue configurations:

- o min-threshold CS1 < max-threshold CS1
- o max-threshold CS1 <= memory assigned to the queue

Note: Many other AQM algorithms exist and are used; they should be configured to achieve a similar result.

5. Additional Information on Service Class Usage

In this section, we provide additional information on how some specific applications should be configured to use the defined service classes.

5.1. Mapping for NTP

From tests that were performed, indications are that precise time distribution requires a very low packet delay variation (jitter) transport. Therefore, we suggest that the following guidelines for Network Time Protocol (NTP) be used:

- o When NTP is used for providing high-accuracy timing within an administrator's (carrier's) network or to end users/clients, the audio service class SHOULD be used, and NTP packets should be marked with EF DSCP value.

- o For applications that require "wall clock" timing accuracy, the Standard service class should be used, and packets should be marked with DF DSCP.

5.2. VPN Service Mapping

"Differentiated Services and Tunnels" [RFC2983] considers the interaction of DiffServ architecture with IP tunnels of various forms. Further to guidelines provided in RFC 2983, below are additional guidelines for mapping service classes that are supported in one part of the network into a VPN connection. This discussion is limited to VPNs that use DiffServ technology for traffic differentiation.

- o The DSCP value(s) that is/are used to represent a PHB or a PHB group SHOULD be the same for the networks at both ends of the VPN tunnel, unless remarking of DSCP is done as ingress/egress processing function of the tunnel. DSCP marking needs to be preserved along the tunnel, end to end.
- o The VPN MAY be configured to support one or more service classes. It is left up to the administrators of the two networks to agree on the level of traffic differentiation that will be provided in the network that supports VPN service. Service classes are then mapped into the supported VPN traffic forwarding behaviors that meet the traffic characteristics and performance requirements of the encapsulated service classes.
- o The traffic treatment in the network that is providing the VPN service needs to be such that the encapsulated service class or classes receive comparable behavior and performance in terms of delay, jitter, and packet loss and that they are within the limits of the service specified.
- o The DSCP value in the external header of the packet forwarded through the network providing the VPN service can be different from the DSCP value that is used end to end for service differentiation in the end network.
- o The guidelines for aggregation of two or more service classes into a single traffic forwarding treatment in the network that is providing the VPN service is for further study.

6. Security Considerations

This document discusses policy and describes a common policy configuration, for the use of a Differentiated Services Code Point by transports and applications. If implemented as described, it should require that the network do nothing that the network has not already allowed. If that is the case, no new security issues should arise from the use of such a policy.

It is possible for the policy to be applied incorrectly, or for a wrong policy to be applied in the network for the defined service class. In that case, a policy issue exists that the network SHOULD detect, assess, and deal with. This is a known security issue in any network dependent on policy-directed behavior.

A well-known flaw appears when bandwidth is reserved or enabled for a service (for example, voice and/or video transport) and another service or an attacking traffic stream uses it. This possibility is inherent in DiffServ technology, which depends on appropriate packet markings. When bandwidth reservation or a priority queuing system is used in a vulnerable network, the use of authentication and flow admission is recommended. To the author's knowledge, there is no known technical way to respond to an unauthenticated data stream using service that it is not intended to use, and such is the nature of the Internet.

The use of a service class by a user is not an issue when the SLA between the user and the network permits him to use it, or to use it up to a stated rate. In such cases, simple policing is used in the Differentiated Services Architecture. Some service classes, such as Network Control, are not permitted to be used by users at all; such traffic should be dropped or remarked by ingress filters. Where service classes are available under the SLA only to an authenticated user rather than to the entire population of users, authentication and authorization services are required, such as those surveyed in [AUTHMECH].

7. Contributing Authors

This section specifically calls out the authors of RFC 4594, from which this document is based on.

Jozef Babiaryz
Nortel Networks

Kwok Ho Chan
Nortel Networks
Email: khchan.work@gmail.com

Fred Baker
Cisco Systems
EMail: fred@cisco.com

Of note, two of the three mentioned authors above worked for Nortel Networks at the time of writing RFC 4594, a company that no longer exists. This author has not seen or heard from those two in many, many years or IETF meetings - as a result of not knowing their new email addresses (or phone numbers).

While much of this document has been rewritten with either edited or

brand new material, there are many short paragraphs that remain as they were from RFC 4594, as well as many sentences that were also left unchanged. Additionally, there were no new graphs, charts, diagrams, or tables introduced, meaning the first 4 tables within this document existed in RFC 4594, created by those authors. Presently, each of those tables contain modified and new information. The last 3 tables, specifically tables 5, 6, & 7 were removed because the examples section was removed.

This author believes there must be proper credit given for all the contributions, including the framework this document retains from that RFC. Periodically, throughout this document, what was written remains the best way of conveying a thought, rule, or otherwise stated behavior or mechanism. Because RFC 4594 was rather large, there is no realistic way of identifying each part that was left untouched. Further, properly quoting that RFC and leaving those sentences embedded in this document would render this document highly unreadable. Another application could be used to show the changes, deletions and additions - but not one that the IETF accepts presently.

This author has created this "Contributing Authors" section as a way of properly identifying those 3 individuals that provided text within this document. We will let the community judge if this is 'good enough' (i.e., rough consensus), or if another way is better.

8. Acknowledgements

The author would like to thank Paul Jones, Glen Lavers, Mo Zanaty, David Benham, Michael Ramalho, Gorrry Fairhurst, David Black, Brian Carpenter, Al Morton, Ruediger Geib and Shitanshu Shah for their comments and questions about this effort that ultimately helped shape this document.

Below are the folks that were acknowledged in RFC 4594, and this author does not want to lose their recognition of contributions to the original effort.

"The authors thank the TSVWG reviewers, David Black, Brian E. Carpenter, and Alan O'Neill for their review and input to this document.

The authors acknowledge a great many inputs, most notably from Bruce Davie, Dave Oran, Ralph Santitoro, Gary Kenward, Francois Audet, Morgan Littlewood, Robert Milne, John Shuler, Nalin Mistry, Al Morton, Mike Pierce, Ed Koehler Jr., Tim Rahrer, Fil Dickinson, Mike Fidler, and Shane Amante. Kimberly King, Joe Zebarth, and Alistair Munroe each did a thorough proofreading,

and the document is better for their contributions."

9. References

9.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC1349] Almquist, P., "Type of Service in the Internet Protocol Suite", RFC 1349, July 1992.
- [RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Service", RFC 2475, December 1998.
- [RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.
- [RFC3246] Davie, B., Charny, A., Bennet, J.C., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, March 2002.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services",

RFC 3662, December 2003.

- [RFC5865] F. Baker, J. Polk, M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, May 2010

9.2. Informative References

- [AUTHMECH] Rescorla, E., "A Survey of Authentication Mechanisms", Work in Progress, September 2005.
- [QBSS] "QBone Scavenger Service (QBSS) Definition", Internet2 Technical Report Proposed Service Definition, March 2001.
- [IEEE1Q] IEEE, 802.1Q Specification
- [IEEE1E] IEEE, 802.1E Wireless LAN User Priority Specification
- [RFC1633] Braden, R., Clark, D., and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.
- [RFC2205] Braden, R., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.
- [RFC2581] Allman, M., Paxson, V., and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.
- [RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, September 1999.
- [RFC2698] Heinanen, J. and R. Guerin, "A Two Rate Three Color Marker", RFC 2698, September 1999.
- [RFC2963] Bonaventure, O. and S. De Cnodder, "A Rate Adaptive Shaper for Differentiated Services", RFC 2963, October 2000.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000.
- [RFC2996] Bernet, Y., "Format of the RSVP DCLASS Object", RFC 2996, November 2000.
- [RFC3086] Nichols, K. and B. Carpenter, "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification", RFC 3086, April 2001.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC

3168, September 2001.

- [RFC3175] Baker, F., Iturralde, C., Le Faucheur, F., and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3175, September 2001.
- [RFC3290] Bernet, Y., Blake, S., Grossman, D., and A. Smith, "An Informal Management Model for Diffserv Routers", RFC 3290, May 2002.
- [RFC3782] Floyd, S., Henderson, T., and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 3782, April 2004.
- [RFC5462] L. Andersson, R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: EXP Field Renamed to Traffic Class Field", RFC 5462, February 2009

Authors' Address

James Polk
3913 Treemont Circle
Colleyville, Texas 76034

Phone: +1.817.271.3552
Email: jmpolk@cisco.com

Appendix A - Changes

Here is a list of all the changes that were captured during the editing process. This will not be a complete list, and others are free to point out what the authors missed, and we'll include that in the next release.

A.1 Since Individual -02 to -03

- o Inserted section 1.6 to explain fundamentally what has changed since RFC 4594, and why changes are necessary.

A.2 Since Individual -01 to -02

- o Added text to the Intro section on the justification from DiffServ Problem Statement draft, as to more of why this update is necessary.
- o Added text to the Intro section expanding on the concept of service classes vs. treatment aggregates (from RFC 5127).

A.3 Since Individual -00 to -01

- o Added Section 2.4 which covers the conflation issues regarding the differences between service classes and treatment aggregates.
- o Added example operational configurations of treatment aggregates applied to this draft's new set of service classes and additional DSCPs.
- o Added references RFC 5865, RFC 5462, IEEE 802.1E and IEEE 802.1Q.

A.4 Since RFC 4594 to Individual Update -00

- o rewrote Intro to emphasize current topics
- o Created a Conversational Service group, comprising the audio, video and Hi-Res service classes - because they have similar characteristics.
- o Incorporated the 6 new DSCPs from [ID-DSCP].
- o moved the example section, en mass, to an appendix that might not be kept for this version. We're not sure it accomplishes what it needs to, and might not provide any real usefulness.
- o Moved 'Realtime-Interactive' service class to CS5, from CS4
- o Changed 'Broadcast Video' service class to 'Broadcast' service class
- o Changed AF4X to 'Video' service class, replacing 'Multimedia Conferencing' service class
- o Moved 'Multimedia Conferencing' service class to different DSCPs
- o Added the 'Hi-Res' service class
- o Removed section 5.1 on signaling choices. It has been included in the main body of the text.
- o Changed document title
- o ...

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2014

R. Stewart
Adara Networks
M. Tuexen
Muenster Univ. of Appl. Sciences
S. Loreto
Ericsson
R. Seggelmann
T-Systems International GmbH
October 20, 2013

A New Data Chunk for Stream Control Transmission Protocol
draft-stewart-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 which causes head of line blocking at the sender side. To overcome this limitation, this document adds a new data chunk to SCTP.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 23, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. N-DATA Chunk	3
3. Procedures	4
4. Socket API Considerations	5
5. IANA Considerations	8
6. Security Considerations	9
7. Acknowledgments	9
8. References	9
Authors' Addresses	10

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 the sender can not interleave different messages.

This document describes a new Data chunk called N-DATA. This chunk incorporates all the flags and properties of the current SCTP Data chunk but also adds a new field in its chunk header, the Fragment Sequence Number (FSN). Then the FSN is only used for reassembly and

the TSN only for the reliability. Therefore, the head of line blocking caused by the original design is avoided.

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. N-DATA Chunk

The following Figure 1 shows the new data chunk N-DATA.

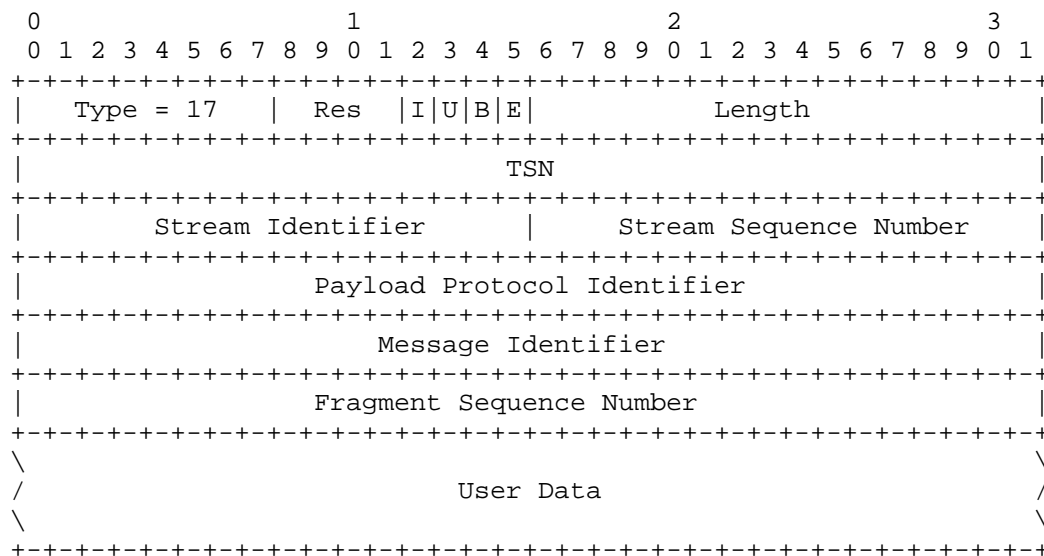


Figure 1: N-DATA chunk format

The only differences between the N-DATA chunk in Figure 1 and the DATA chunk defined in [RFC4960] and [I-D.ietf-tsvwg-sctp-sack-immediately] is the addition of the new Message Identifier (MID) and Fragment Sequence Number (FSN).

Message Identifier (MID): 32 bits (unsigned integer)

The Message Identifier . Please note that the MID is in "network byte order", a.k.a. Big Endian.

Fragment Sequence Number (FSN): 32 bits (unsigned integer)

Identifies the fragment number of this piece of a message. FSN's are unsigned number, the first fragment MUST start at 0 and MUST have the 'B' bit set. The last fragment of a message MUST have

the 'E' bit set. Note that the FSN may wrap completely multiple times allowing arbitrary large messages. Please note that the FSN is in "network byte order", a.k.a. Big Endian.

3. Procedures

3.1. Sender Side Considerations

A sender MUST NOT send a N-DATA chunk unless the peer has indicated its support of the N-DATA chunk type within the Supported Extensions Parameter as defined in [RFC5061].

A sender MUST NOT use the N-DATA chunk unless the user has requested that use 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 N-DATA chunk, the key is that the the stack MUST know the application has indicated its choice in wanting to use the extension.

Sender side usage of the N-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. If the 'I' bit is set the 'E' bit MUST also be set, i.e. the 'I' bit may only be set on the last fragment of a message. All other properties of the existing SCTP DATA chunk also apply to the N-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.

3.2. Receiver Side Considerations

Upon reception of an SCTP packet containing a N-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.

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. Socket Options

option name	data type	get	set
SCTP_NDATA_ENABLE	int	X	X
SCTP_PLUGGABLE_SS	struct sctp_assoc_value	X	X
SCTP_SS_VALUE	struct sctp_stream_value	X	X

4.1.1. Enable or Disable the Interleaving Capability (SCTP_NDATA_ENABLE)

A new socket option to turn on/off the usage of the N-DATA chunk. Turning this option on only effect future associations, and MUST be turned on for the protocol stack to indicate support of the N-DATA chunk to the peer during association setup. Turning this option off, will prevent the N-DATA chunk from being indicated supported in future associations, and will also prevent current associations from producing N-DATA chunks for future large fragmented messages. Note that this does not stop the peer from sending N-DATA chunks.

An N-DATA chunk aware application 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.

4.1.1.2. 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]:

```
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_BANDWIDTH`: 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_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.

4.1.3. 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:

name	per stream info
SCTP_SS_DEFAULT	no
SCTP_SS_RR	no
SCTP_SS_RR_INTER	no
SCTP_SS_RR_PKT	no
SCTP_SS_RR_PKT_INTER	no
SCTP_SS_PRIO	yes
SCTP_SS_PRIO_INTER	yes
SCTP_SS_FB	no
SCTP_SS_FB_INTER	no
SCTP_SS_FCFS	no

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:

```
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 '00'. This requires an additional line in the "Chunk Types" registry for SCTP:

ID Value	Chunk Type	Reference
17	New DATA chunk (N-DATA)	[RFCXXXX]

The registration table as defined in [RFC6096] for the chunk flags of this chunk type is initially given by the following table:

Chunk Flag Value	Chunk Flag Name	Reference
0x01	E bit	[RFCXXXX]
0x02	B bit	[RFCXXXX]
0x04	U bit	[RFCXXXX]
0x08	I bit	[RFCXXXX]

0x10	Unassigned		
0x20	Unassigned		
0x40	Unassigned		
0x80	Unassigned		
+-----+	+-----+	+-----+	+-----+

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 Lixia Zhang for her invaluable comments.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.
- [RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, January 2011.
- [I-D.ietf-tsvwg-sctp-sack-immediately] Tuexen, M., Ruengeler, I., and R. Stewart, "SACK-IMMEDIATELY Extension for the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-sack-immediately-04 (work in progress), August 2013.

8.2. Informative References

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

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V.
Yasevich, "Sockets API Extensions for the Stream Control
Transmission Protocol (SCTP)", RFC 6458, December 2011.

Authors' Addresses

Randall R. Stewart
Adara Networks
Chapin, SC 29036
US

Email: randall@lakerest.net

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

Email: tuexen@fh-muenster.de

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: Salvatore.Loreto@ericsson.com

Robin Seggelmann
T-Systems International GmbH
Fasanenweg 5
70771 Leinfelden-Echterdingen
DE

Email: robin.seggelmann@t-systems.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 11, 2014

M. Thornburgh
Adobe
July 10, 2013

Adobe's Secure Real-Time Media Flow Protocol
draft-thornburgh-adobe-rtmfp-10

Abstract

This memo describes Adobe's Secure Real-Time Media Flow Protocol (RTMFP), an endpoint-to-endpoint communication protocol designed to securely transport parallel flows of real-time video, audio, and data messages, as well as bulk data, over IP networks. RTMFP has features making it effective for peer-to-peer (P2P) as well as client-server communications, even when Network Address Translators (NATs) are used.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

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

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

This Internet-Draft will expire on January 11, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	5
1.1. Design Highlights of RTMFP	6
1.2. Terminology	7
2. Syntax	7
2.1. Common Elements	8
2.1.1. Elementary Types and Constructs	8
2.1.2. Variable Length Unsigned Integer (VLU)	9
2.1.3. Option	10
2.1.4. Option List	11
2.1.5. Internet Socket Address (Address)	12
2.2. Network Layer	13
2.2.1. Encapsulation	13
2.2.2. Multiplex	13
2.2.3. Encryption	14
2.2.4. Packet	15
2.3. Chunks	18
2.3.1. Packet Fragment Chunk	20
2.3.2. Initiator Hello Chunk (IHello)	21
2.3.3. Forwarded Initiator Hello Chunk (FIHello)	21
2.3.4. Responder Hello Chunk (RHello)	22
2.3.5. Responder Redirect Chunk (Redirect)	23
2.3.6. RHello Cookie Change Chunk	25
2.3.7. Initiator Initial Keying Chunk (IIKeying)	26
2.3.8. Responder Initial Keying Chunk (RIKeying)	27
2.3.9. Ping Chunk	29
2.3.10. Ping Reply Chunk	30
2.3.11. User Data Chunk	30
2.3.11.1. Options for User Data	32
2.3.11.1.1. User's Per-Flow Metadata	33
2.3.11.1.2. Return Flow Association	33
2.3.12. Next User Data Chunk	34
2.3.13. Data Acknowledgement Bitmap Chunk (Bitmap Ack)	36
2.3.14. Data Acknowledgement Ranges Chunk (Range Ack)	38
2.3.15. Buffer Probe Chunk	40
2.3.16. Flow Exception Report Chunk	41
2.3.17. Session Close Request Chunk (Close)	42
2.3.18. Session Close Acknowledgement Chunk (Close Ack)	42
3. Operation	42
3.1. Overview	43

3.2.	Endpoint Identity	44
3.3.	Packet Multiplex	46
3.4.	Packet Fragmentation	46
3.5.	Sessions	48
3.5.1.	Startup	50
3.5.1.1.	Normal Handshake	50
3.5.1.1.1.	Initiator	51
3.5.1.1.2.	Responder	53
3.5.1.2.	Cookie Change	55
3.5.1.3.	Glare	57
3.5.1.4.	Redirector	58
3.5.1.5.	Forwarder	59
3.5.1.6.	Redirector and Forwarder with NAT	61
3.5.1.7.	Load Distribution and Fault Tolerance	64
3.5.2.	Congestion Control	65
3.5.2.1.	Time Critical Reverse Notification	66
3.5.2.2.	Retransmission Timeout	66
3.5.2.3.	Burst Avoidance	68
3.5.3.	Address Mobility	69
3.5.4.	Ping	69
3.5.4.1.	Keepalive	70
3.5.4.2.	Address Mobility	70
3.5.4.3.	Path MTU Discovery	71
3.5.5.	Close	71
3.6.	Flows	72
3.6.1.	Overview	72
3.6.1.1.	Identity	73
3.6.1.2.	Messages and Sequencing	73
3.6.1.3.	Lifetime	74
3.6.2.	Sender	75
3.6.2.1.	Startup	77
3.6.2.2.	Queuing Data	78
3.6.2.3.	Sending Data	78
3.6.2.3.1.	Startup Options	80
3.6.2.3.2.	Send Next Data	80
3.6.2.4.	Processing Acknowledgements	81
3.6.2.5.	Negative Acknowledgement and Loss	81
3.6.2.6.	Timeout	82
3.6.2.7.	Abandoning Data	83
3.6.2.7.1.	Forward Sequence Number Update	83
3.6.2.8.	Examples	84
3.6.2.9.	Flow Control	85
3.6.2.9.1.	Buffer Probe	86
3.6.2.10.	Exception	86
3.6.2.11.	Close	86
3.6.3.	Receiver	87
3.6.3.1.	Startup	89
3.6.3.2.	Receiving Data	90

3.6.3.3.	Buffering and Delivering Data	92
3.6.3.4.	Acknowledging Data	94
3.6.3.4.1.	Timing	94
3.6.3.4.2.	Size and Truncation	95
3.6.3.4.3.	Constructing	96
3.6.3.4.4.	Delayed Acknowledgement	96
3.6.3.4.5.	Obligatory Acknowledgement	96
3.6.3.4.6.	Opportunistic Acknowledgement	97
3.6.3.4.7.	Example	97
3.6.3.5.	Flow Control	98
3.6.3.6.	Receiving a Buffer Probe	99
3.6.3.7.	Rejecting a Flow	100
3.6.3.8.	Close	100
4.	IANA Considerations	101
5.	Security Considerations	101
6.	Acknowledgements	103
7.	References	103
7.1.	Normative References	103
7.2.	Informative References	103
Appendix A.	Example Congestion Control Algorithm	104
A.1.	Discussion	104
A.2.	Algorithm	105
Author's Address	108

1. Introduction

Adobe's Secure Real-Time Media Flow Protocol (RTMFP) is intended for use as a general purpose endpoint-to-endpoint data transport service in IP networks. It has features that make it well suited to the transport of real-time media (such as low-delay video, audio, and data) as well as bulk data, and for client-server as well as peer-to-peer (P2P) communication. These features include independent parallel message flows which may have different delivery priorities, variable message reliability (from TCP-like full reliability to UDP-like best effort), multi-point congestion control, and built-in security. Session multiplexing and facilities to support UDP hole-punching simplify Network Address Translator (NAT) traversal in peer-to-peer systems.

RTMFP is implemented in Flash Player, Adobe Integrated Runtime (AIR), and Adobe Media Server (AMS, formerly Flash Media Server or FMS), all from Adobe Systems Incorporated, and is used as the foundation transport protocol for real-time video, audio, and data communication, both client-server and P2P, in those products. At the time of writing, the Adobe Flash Player runtime is installed on more than one billion end-user desktop computers.

RTMFP was developed by Adobe Systems Incorporated, and is not the product of an IETF activity.

This memo describes the syntax and operation of the Secure Real-Time Media Flow Protocol.

This memo describes a general security framework that, when combined with an application-specific Cryptography Profile, can be used to establish a confidential and authenticated session between endpoints. The application-specific Cryptography Profile, not defined herein, would detail the specific cryptographic algorithms, data formats, and semantics to be used within this framework. Interoperation between applications of RTMFP requires common or compatible Cryptography Profiles.

Note to implementers: at the time of writing, the Cryptography Profile used by the above mentioned Adobe products is not publicly described by Adobe. Implementers should investigate the availability of documentation of that Cryptography Profile prior to implementing RTMFP for the purpose of interoperation with the above mentioned Adobe products.

1.1. Design Highlights of RTMFP

Between any pair of communicating endpoints is a single, bidirectional, secured, congestion controlled session. Unidirectional flows convey messages from one end to the other within the session. An endpoint can have concurrent sessions with multiple other far endpoints.

Design highlights of RTMFP include:

- o The security framework is an inherent part of the basic protocol. The application designer chooses the cryptographic formats and algorithms to suit the needs of the application, and may update them as the state of the security arts progresses.
- o Cryptographic Endpoint Discriminators can resist port scanning.
- o All header, control, and framing information, except for network addressing information and a session identifier, is encrypted according to the Cryptography Profile.
- o There is a single session and associated congestion control state between a pair of endpoints.
- o Each session may have zero or more unidirectional message-oriented flows in each direction. All of a session's sending flows share the session's congestion control state.
- o Return Flow Association (Section 2.3.11.1.2) generalizes bidirectional communication to arbitrarily complex trees of flows.
- o Messages in flows can be arbitrarily large and are fragmented for transmission.
- o Messages of any size may be sent with full, partial, or no reliability (sender's choice). Messages may be delivered to the receiving user in original queuing order or network arrival order (receiver's choice).
- o Flows are named with arbitrary, user-defined metadata (Section 2.3.11.1.1) rather than port or stream numbers.
- o The sequence numbers of each flow are independent of all other flows, and are not permanently bound to a session-wide transmission ordering. This allows real-time priority decisions to be made at transmission or retransmission time.

- o Each flow has its own receive window, and therefore independent flow control.
- o Round-trips are expensive, and are minimized or eliminated when possible.
- o After a session is established, flows begin by sending the flow's messages with no additional handshake (and associated round-trips).
- o Transmitting bytes on the network is much more expensive than moving bytes in a CPU or memory. Wasted bytes are minimized or eliminated when possible and practical, and variable length encodings are used, even at the expense of breaking 32-bit alignment and making the text diagrams in this specification look awkward.
- o P2P lookup and peer introduction (including UDP hole punching for NAT and firewall traversal) is supported directly by the session startup handshake.
- o Session identifiers allow an endpoint to multiplex many sessions over a single local transport address while allowing sessions to survive changes in transport address (as may happen in mobile or wireless deployments).

The syntax of the protocol is detailed in Section 2. The operation of the protocol is detailed in Section 3.

1.2. Terminology

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

2. Syntax

Definitions of types and structures in this specification use traditional text diagrams paired with procedural descriptions using a C-like syntax. The C-like procedural descriptions SHALL be construed as definitive.

Structures are packed to take only as many bytes as explicitly indicated. There is no 32-bit alignment constraint, and fields are not padded for alignment unless explicitly indicated or described. Text diagrams may include a bit ruler across the top; this is a

convenience for counting bits in individual fields and does not necessarily imply field alignment on a multiple of the ruler width.

Unless specified otherwise, reserved fields SHOULD be set to 0 by a sender and MUST be ignored by a receiver.

The procedural syntax of this specification defines correct and error-free encoded inputs to a parser. The procedural syntax does not describe a fully featured parser, including error detection and handling. Implementations MUST include means to identify error circumstances, including truncations causing elementary or composed types to not fit inside containing structures, fields or elements. Unless specified otherwise, an error circumstance SHALL abort the parsing and processing of an element and its enclosing elements, up to the containing packet.

2.1. Common Elements

This section lists types and structures that are used throughout this specification.

2.1.1. Elementary Types and Constructs

This section lists the elementary types and constructs out of which all of the following sections' definitions are built.

`uint8_t var;`

An unsigned integer 8 bits (one byte) in length and byte aligned.

`uint16_t var;`

An unsigned integer 16 bits in length, in network byte order ("big endian") and byte aligned.

`uint32_t var;`

An unsigned integer 32 bits in length, in network byte order and byte aligned.

`uint128_t var;`

An unsigned integer 128 bits in length, in network byte order and byte aligned.

`uintn_t var :bitsize;`

An unsigned integer of any other size, potentially not byte aligned.

Its size in bits is specified explicitly by bitsize.

```
bool_t var :1;
```

A boolean flag having the value true (1 or set) or false (0 or clear) and being one bit in length.

```
type var[num];
```

A packed array of type with length num*sizeof(type)*8 bits.

```
struct name_t { ... } name :bitsize;
```

A packed structure. Its size in bits is specified by bitsize.

```
remainder();
```

The number of bytes from the current offset to the end of the enclosing structure.

```
type var[remainder()];
```

A packed array of type, its size extending to the end of the enclosing structure.

Note that a bitsize of "variable" indicates that the size of the structure is determined by the sizes of its interior components. A bitsize of "n*8" indicates that the size of the structure is a whole number of bytes and is byte aligned.

2.1.2. Variable Length Unsigned Integer (VLU)

A VLU encodes any finite non-negative integer into one or more bytes. For each encoded byte, if the high bit is set, the next byte is also part of the VLU. If the high bit is clear, this is the final byte of the VLU. The remaining bits encode the number, seven bits at a time, from most significant to least significant.

```

      0 1 2 3 4 5 6 7
+~+~+~+~+~+~+~+
|1|   digit   |.....|0|   digit   |
+~+~+~+~+~+~+~+
^
+----- zero or more -----+

```

```

struct vlu_t
{
    value = 0;
    do {
        bool_t more :1;
        uintn_t digit :7;
        value = (value * 128) + digit;
    } while(more);
} :variable*8;

```

```

+-----/-+
|           \ |
+-----/-+

```

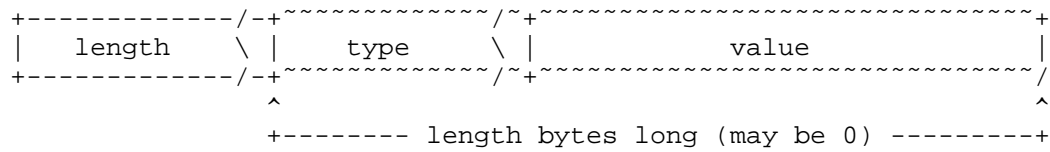
Figure 1: VLU depiction in following diagrams

Unless stated otherwise in this specification, implementations SHOULD handle VLUs encoding unsigned integers at least 64 bits in length (that is, encoding a maximum value of at least $2^{64} - 1$).

2.1.3. Option

An Option is a Length-Type-Value triplet. Length and Type are encoded in VLU format. Length is the number of bytes of payload following the Length field. The payload comprises the Type and Value fields. Type identifies the kind of option this is. The syntax of the Value field is determined by the type of option.

An option can have a length of zero, in which case it has no type and no value and is empty. An empty option is called a "Marker".



```

struct option_t
{
    vlu_t length :variable*8; // "L"
    if(length > 0)
    {
        struct {
            vlu_t type :variable*8; // "T"
            uint8_t value[remainder()]; // "V"
        } payload :length*8;
    }
} :variable*8;

```

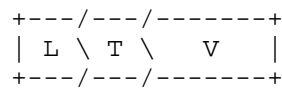
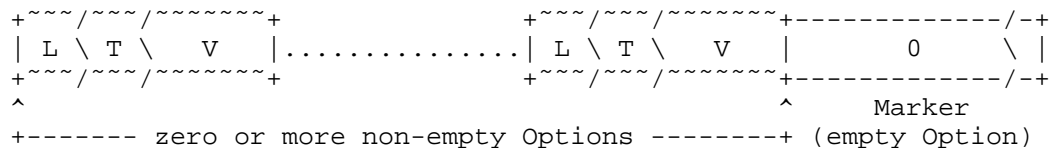


Figure 2: Option depiction in following diagrams

2.1.4. Option List

An Option List is a sequence of zero or more non-empty Options terminated by a Marker.



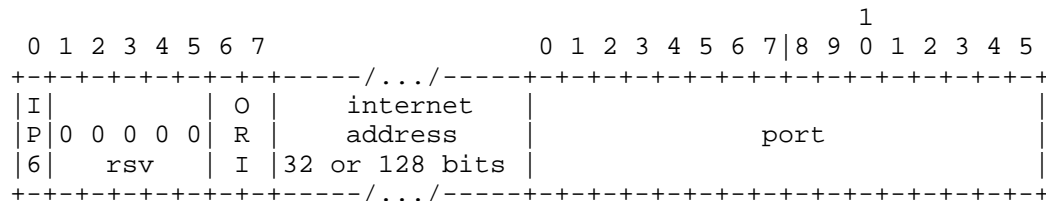
```

struct optionList_t
{
    do
    {
        option_t option :variable*8;
    } while(option.length > 0);
} :variable*8;

```

2.1.5. Internet Socket Address (Address)

When communicating an Internet Socket Address (a combination of a 32-bit IPv4 [RFC0791] or 128-bit IPv6 [RFC2460] address and a 16-bit port number) to another RTMFP, this encoding is used. This encoding additionally allows an address to be tagged with an origin type, which an RTMFP MAY use to modify the use or disposition of the address.



```
struct address_t
{
    bool_t    inet6      :1;        // "IP6"
    uintn_t   reserved   :5 = 0;    // "rsv"
    uintn_t   origin     :2;        // "ORI"
    if(inet6)
        uint128_t ipAddress;
    else
        uint32_t  ipAddress;
        uint16_t  port;
} :variable*8;
```

inet6 : If set, the Internet address is a 128-bit IPv6 address. If clear, the Internet address is a 32-bit IPv4 address.

origin : The origin tag of this address. Possible values are:

- ```

0 : Unknown, unspecified, or "other"

1 : Address was reported by the origin as a local, directly-
 attached interface address

2 : Address was observed to be the source address from which a
 packet was received (a "reflexive transport address" in the
 terminology of [RFC5389])

3 : Address is a relay, proxy, or introducer (a Redirector and/or
 Forwarder)

```

ipAddress : The Internet address, in network byte order.

port : The 16 bit port number, in network byte order.

## 2.2. Network Layer

### 2.2.1. Encapsulation

RTMFP Multiplex packets are usually carried in UDP [RFC0768] datagrams so that they may transit commonly deployed NATs and firewalls, and so that RTMFP may be implemented on commonly deployed operating systems without special privileges or permissions.

RTMFP Multiplex packets MAY be carried by any suitable datagram transport or encapsulation where endpoints are addressed by an Internet socket address (that is, an IPv4 or IPv6 address and a 16-bit port number).

The choice of port numbers is not mandated by this specification. Higher protocol layers or the application define the port numbers used.

### 2.2.2. Multiplex

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| scrambled session ID (SSID) |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| e first32[0] |
| - - - - - n - |
| c first32[1] |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| r - |
| y - |
| pted packet - |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

struct multiplex_t
{
 uint32_t scrambledSessionID; // "SSID"
 union {
 uint32_t first32[2]; // see note
 uint8_t encryptedPacket[remainder()];
 } :(encapsulation.length - 4)*8;

 // if encryptedPacket is less than 8 bytes long, treat it
 // as if it were end-padded with 0s for the following:
 sessionID = scrambledSessionID XOR first32[0] XOR first32[1];
} :encapsulation.length*8;

```

The 32-bit Scrambled Session ID is the 32-bit Session ID modified by performing a bitwise exclusive-or with the bitwise exclusive-or of the first two 32-bit words of the encrypted packet.

The Session ID is a 32-bit value that the receiver has requested to be used by the sender when sending packets to this receiver (Section 2.3.7, Section 2.3.8). The Session ID identifies the Session to which this packet belongs and the decryption key to be used to decrypt the encrypted packet.

Note: Session ID 0 (prior to scrambling) denotes the startup pseudo-session and implies the Default Session Key.

Note: If the encrypted packet is less than 8 bytes long, then for the scrambling operation, perform the exclusive-or as though the encrypted packet were end-padded with enough 0-bytes to bring its length to 8.

### 2.2.3. Encryption

RTMFP packets are encrypted according to a Cryptography Profile. This specification doesn't define a Cryptography Profile or mandate a particular choice of cryptography. The application defines the cryptographic syntax and algorithms.

Packet encryption is RECOMMENDED to be a block cipher operating in Cipher Block Chaining [CBC] or similar mode. Encrypted packets MUST be decipherable without inter-packet dependency, since packets may be lost, duplicated, or reordered in the network.

The packet encryption layer is responsible for data integrity and authenticity of packets, for example by means of a checksum or cryptographic message authentication code. To mitigate replay attacks, data integrity SHOULD comprise duplicate packet detection, for example by means of a session-wide packet sequence number. The packet encryption layer SHALL discard a received packet that does not pass integrity or authenticity tests.

Note that the structures described below are of plain, unencrypted packets. Encrypted packets MUST be decrypted according to the Session Key associated with the Multiplex Session ID before being interpreted according to this specification.

The cryptography profile defines a well-known Default Session Key that is used at session startup, during which per-session key(s) are negotiated by the two endpoints. A Session ID of zero denotes use of the Default Session Key. The Default Session Key is also used with non-zero Session IDs during the latter phases of session startup

(Section 2.3.6, Section 2.3.8). See Security Considerations (Section 5) for more about the Default Session Key.

#### 2.2.4. Packet

An (unencrypted, plain) RTMFP Packet consists of a variable sized common header, zero or more chunks, and padding. Padding can be inserted by the encryption layer of the sender to meet cipher block size constraints, and is ignored by the receiver. A sender's encryption layer MAY pad the end of a packet with bytes with value 0xff such that the resulting packet is a natural and appropriate size for the cipher. Alternatively, the Cryptography Profile MAY define its own framing and padding scheme, if needed, such that decrypted packets are compatible with the syntax defined in this section.

[illegible]

```

struct packet_t
{
 bool_t timeCritical :1; // "TC"
 bool_t timeCriticalReverse :1; // "TCR"
 uintn_t reserved :2; // "rsv"
 bool_t timestampPresent :1; // "TS"
 bool_t timestampEchoPresent :1; // "TSE"
 uintn_t mode :2; // "MOD"
 if(0 != mode)
 {
 if(timestampPresent)
 uint16_t timestamp;
 if(timestampEchoPresent)
 uint16_t timestampEcho;
 while(remainder() > 2)
 {
 uint8_t chunkType;
 uint16_t chunkLength;
 if(remainder() < chunkLength)
 break;
 uint8_t chunkPayload[chunkLength];
 } // chunks
 uint8_t padding[remainder()];
 }
} :plainPacket.length*8;

```

timeCritical : Time Critical Forward Notification. If set, indicates that this packet contains real-time user data.

timeCriticalReverse : Time Critical Reverse Notification. If set, indicates that the sender is currently receiving packets on other sessions that have the timeCritical flag set.

timestampPresent : If set, indicates that the timestamp field is present. If clear, there is no timestamp field.

timestampEchoPresent : If set, indicates that the timestamp echo field is present. If clear, there is no timestamp echo field.

mode : The mode of this packet. See below for additional discussion of packet modes. Possible values are:

0 : Forbidden value

1 : Initiator Mark

2 : Responder Mark

3 : Startup

timestamp : If the timestampPresent flag is set, this field is present and contains the low 16 bits of the sender's 250 Hz clock (4 milliseconds per tick) at transmit time. The sender's clock MAY have its origin at any time in the past.

timestampEcho : If the timestampEchoPresent flag is set, this field is present and contains the sender's estimate of what the timestamp field of a packet received from the other end would be at the time this packet was transmitted, using the method described in Section 3.5.2.2.

chunks : Zero or more chunks follow the header. It is RECOMMENDED that a packet contain at least one chunk.

padding : Zero or more bytes of padding follow the chunks. The following conditions indicate padding:

- \* Fewer than three bytes (the size of a chunk header) remain in the packet.

- \* The chunkLength field of what would be the current chunk header indicates that the hypothetical chunk payload wouldn't fit in the remaining bytes of the packet.

Packet mode 0 is not allowed. Packets marked with this mode are invalid and MUST be discarded.

The original initiator of a session MUST mark all non-startup packets it sends in that session with packet mode 1 "Initiator Mark". It SHOULD ignore any packet received in that session with packet mode 1.

The original responder of a session MUST mark all non-startup packets it sends in that session with packet mode 2 "Responder Mark". It SHOULD ignore any packet received in that session with packet mode 2.

Packet mode 3 is for session startup. Session startup chunks are only allowed in packets with this mode.

Chunks that are not for session startup are only allowed in packets with modes 1 or 2.

### 2.3. Chunks

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| chunkType | chunkLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+-----+
| chunkPayload (chunkLength bytes, may be zero) |
+-----+

```

```

struct chunk_t
{
 uint8_t chunkType;
 uint16_t chunkLength;
 uint8_t chunkPayload[chunkLength];
} :variable*8;

```

chunkType : The chunk type code.

chunkLength : The size, in bytes, of the chunk payload.

chunkPayload : The type-specific payload of this chunk, chunkLength bytes in length (may be empty).

Defined chunk types are enumerated here in the order they might be encountered in the course of a typical session. The following chunk type codes are defined:



0x7f : Packet Fragment (Section 2.3.1)  
0x30 : Initiator Hello (Section 2.3.2)  
0x0f : Forwarded Initiator Hello (Section 2.3.3)  
0x70 : Responder Hello (Section 2.3.4)  
0x71 : Responder Redirect (Section 2.3.5)  
0x79 : RHello Cookie Change (Section 2.3.6)  
0x38 : Initiator Initial Keying (Section 2.3.7)  
0x78 : Responder Initial Keying (Section 2.3.8)  
0x01 : Ping (Section 2.3.9)  
0x41 : Ping Reply (Section 2.3.10)  
0x10 : User Data (Section 2.3.11)  
0x11 : Next User Data (Section 2.3.12)  
0x50 : Data Acknowledgement Bitmap (Section 2.3.13)  
0x51 : Data Acknowledgement Ranges (Section 2.3.14)  
0x18 : Buffer Probe (Section 2.3.15)  
0x5e : Flow Exception Report (Section 2.3.16)  
0x0c : Session Close Request (Section 2.3.17)  
0x4c : Session Close Acknowledgement (Section 2.3.18)  
0x00 : Ignore/Padding  
0xff : Ignore/Padding

A receiver MUST ignore a chunk having an unrecognized chunk type code. A receiver MUST ignore a chunk appearing in a packet having a mode inappropriate to that chunk type.

Unless specified otherwise, if a chunk has a syntax or processing error (for example, the chunk's payload field is not long enough to contain the specified syntax elements), the chunk SHALL be ignored as though it was not present in the packet, and parsing and processing

SHALL commence with the next chunk in the packet, if any.

### 2.3.1. Packet Fragment Chunk

This chunk is used to divide a plain RTMFP packet (Section 2.2.4) that is unavoidably larger than the path MTU (such as session startup packets containing Responder Hello (Section 2.3.4) or Initiator Initial Keying (Section 2.3.7) chunks with large certificates) into segments that do not exceed the path MTU, and to allow the segments to be sent through the network at a moderated rate to avoid jamming interfaces, links, or paths.

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|M| 0x7f | chunkLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|M| reserved | packetID \ | fragmentNum \ |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+-----+-----+-----+-----+-----+-----+-----+-----+
| packetFragment
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

struct fragmentChunkPayload_t
{
 bool_t moreFragments :1; // M
 uintn_t reserved :7;
 vlu_t packetID :variable*8;
 vlu_t fragmentNum :variable*8;
 uint8_t packetFragment[remainder()];
} :chunkLength*8;

```

**moreFragments** : If set, the indicated packet comprises additional fragments. If clear, this fragment is the final fragment of the packet.

**reserved** : Reserved for future use.

**packetID** : VLU, the identifier of this segmented packet. All fragments of the same packet have the same packetID.

**fragmentNum** : VLU, the index of this fragment of the indicated packet. The first fragment of the packet MUST be index 0. Fragments are numbered consecutively.

packetFragment : The bytes of the indicated segment of the indicated original plain RTMFP packet. A packetFragment MUST NOT be empty.

The use of this mechanism is detailed in Section 3.4.

### 2.3.2. Initiator Hello Chunk (IHello)

This chunk is sent by the initiator of a new session to begin the startup handshake. This chunk is only allowed in a packet with Session ID 0, encrypted with the default session key, and having packet mode 3 (Startup).

```

0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x30 | chunkLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+-----/-----+-----+-----+-----+-----+-----+-----+
| epdLength \ | endpointDiscriminator (epdLength bytes) |
+-----/-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
| tag |
+-----+-----+-----+-----+-----+-----+-----+

```

```

struct ihelloChunkPayload_t
{
 vlu_t epdLength :variable*8;
 uint8_t endpointDiscriminator[epdLength];
 uint8_t tag[remainder()];
} :chunkLength*8;

```

epdLength : VLU, the length of the following endpointDiscriminator field in bytes.

endpointDiscriminator : The Endpoint Discriminator for the identity with which the initiator wants to communicate.

tag : Initiator-provided data to be returned in a Responder Hello's tagEcho field. The tag/tagEcho is used to match Responder Hellos to the initiator's session startup state independent of the responder's address.

The use of IHello is detailed in Section 3.5.1.

### 2.3.3. Forwarded Initiator Hello Chunk (FIHello)

This chunk is sent on behalf of an initiator by a Forwarder. It is only allowed in packets of an established session having packet mode 1 or 2. A receiver MAY treat this chunk as though it was an

Initiator Hello received directly from replyAddress. Alternatively, if the receiver is selected by the Endpoint Discriminator, it MAY respond to replyAddress with an Implied Redirect (Section 2.3.5).

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+-----+-----+-----+-----+
| 0x0f | chunkLength |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| epdLength \ | endpointDiscriminator (epdLength bytes) |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| replyAddress |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| tag |
+-----+-----+-----+-----+

```

```

struct fihelloChunkPayload_t
{
 vlu_t epdLength :variable*8;
 uint8_t endpointDiscriminator[epdLength];
 address_t replyAddress :variable*8;
 uint8_t tag[remainder()];
} :chunkLength*8;

```

epdLength : VLU, the length of the following endpointDiscriminator field in bytes.

endpointDiscriminator : The Endpoint Discriminator for the identity with which the original initiator wants to communicate, copied from the original Initiator Hello.

replyAddress : Address format (Section 2.1.5), the address that the forwarding node derived from the received Initiator Hello, to which the receiver should respond.

tag : Copied from the original Initiator Hello.

The use of FIHello is detailed in Section 3.5.1.5.

#### 2.3.4. Responder Hello Chunk (RHello)

This chunk is sent by a responder in response to an Initiator Hello or Forwarded Initiator Hello if the Endpoint Discriminator indicates the responder's identity. This chunk is only allowed in a packet with Session ID 0, encrypted with the default session key, and having packet mode 3 (Startup).

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x70 | chunkLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+-----/-----+-----+-----+-----+-----+-----+-----+
| tagLength \ | tagEcho (tagLength bytes) |
+-----/-----+-----+-----+-----+-----+-----+-----+
+-----/-----+-----+-----+-----+-----+-----+-----+
| cookieLength\ | cookie (cookieLength bytes) |
+-----/-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
| responderCertificate
+-----+-----+-----+-----+-----+-----+-----+

```

```

struct rhelloChunkPayload_t
{
 vlu_t tagLength :variable*8;
 uint8_t tagEcho[tagLength];
 vlu_t cookieLength :variable*8;
 uint8_t cookie[cookieLength];
 uint8_t responderCertificate[remainder()];
} :chunkLength*8;

```

tagLength : VLU, the length of the following tagEcho field in bytes.

tagEcho : The tag from the Initiator Hello, unaltered.

cookieLength : VLU, the length of the following cookie field in bytes.

cookie : Responder-created state data to authenticate a future Initiator Initial Keying message (in order to prevent denial of service attacks).

responderCertificate : The responder's cryptographic credentials.

Note: this specification doesn't mandate a specific choice of certificate format. The Cryptography Profile determines the syntax, algorithms, and interpretation of the responderCertificate.

The use of RHello is detailed in Section 3.5.1.

#### 2.3.5. Responder Redirect Chunk (Redirect)

This chunk is sent in response to an Initiator Hello or Forwarded Initiator Hello to indicate that the requested endpoint can be reached at one or more of the indicated address(es). A receiver can add none, some, or all of the indicated address(es) to the set of

addresses to which it is sending Initiator Hello messages for the opening session associated with tagEcho. This chunk is only allowed in a packet with Session ID 0, encrypted with the default session key, and having packet mode 3 (Startup).

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x71 | chunkLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+-----+-----+-----+-----+-----+-----+-----+-----+
| tagLength \ | tagEcho (tagLength bytes) |
+-----+-----+-----+-----+-----+-----+-----+-----+
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+
| redirectDestination 1 |
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+
:
:
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+
| redirectDestination N |
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+

```

```

struct responderRedirectChunkPayload_t
{
 vlu_t tagLength :variable*8;
 uint8_t tagEcho[tagLength];
 addressCount = 0;
 while(remainder() > 0)
 {
 address_t redirectDestination :variable*8;
 addressCount++;
 }
 if(0 == addressCount)
 redirectDestination = packetSourceAddress();
} :chunkLength*8;

```

tagLength : VLU, the length of the following tagEcho field in bytes.

tagEcho : The tag from the Initiator Hello, unaltered.

redirectDestination : (Zero or more) Address format (Section 2.1.5), addresses to add to the opening set for the indicated session.

If this chunk lists zero redirectDestination addresses, then this is an Implied Redirect, and the indicated address is the address from which the packet containing this chunk was received.

The use of Redirect is detailed in Section 3.5.1.1.1, Section 3.5.1.1.2, and Section 3.5.1.4.

### 2.3.6. RHello Cookie Change Chunk

This chunk SHOULD be sent by a responder to an initiator in response to an Initiator Initial Keying if that chunk's cookie appears to have been created by the responder but the cookie is incorrect (for example, it includes a hash of the initiator's address, but the initiator's address is different than the one which elicited the Responder Hello containing the original cookie).

This chunk is only allowed in a packet encrypted with the default session key and having packet mode 3, and with the Session ID indicated in the initiatorSessionID field of the Initiator Initial Keying to which this is a response.

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0x79 | chunkLength |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| oldCookieLen\ | oldCookie (oldCookieLen bytes) |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| newCookie |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

struct rhelloCookieChangeChunkPayload_t
{
 vlu_t oldCookieLen :variable*8;
 uint8_t oldCookie[oldCookieLen];
 uint8_t newCookie[remainder()];
} :chunkLength*8;

```

oldCookieLen : VLU, the length of the following oldCookie field in bytes.

oldCookie : The cookie that was sent in a previous Responder Hello and Initiator Initial Keying.

newCookie : The new cookie that the responder would like sent (and signed) in a replacement Initiator Initial Keying. The old and new cookies need not have the same lengths.

On receipt of this chunk, the initiator SHOULD compute, sign, and send a new Initiator Initial Keying having newCookie in place of oldCookie. The use of this chunk is detailed in Section 3.5.1.2.

## 2.3.7. Initiator Initial Keying Chunk (IIKeying)

This chunk is sent by an initiator to establish a session with a responder. The initiator MUST have obtained a valid cookie to use with the responder, typically by receiving a Responder Hello from it. This chunk is only allowed in a packet with Session ID 0, encrypted with the default session key, and having packet mode 3 (Startup).



```
struct iikeyingChunkPayload_t
{
```

```
 struct
 {
 uint32_t initiatorSessionID;
 vlu_t cookieLength :variable*8;
 uint8_t cookieEcho[cookieLength];
 vlu_t certLength :variable*8;
 uint8_t initiatorCertificate[certLength];
 vlu_t skicLength :variable*8;
 uint8_t sessionKeyInitiatorComponent[skicLength];
 } initiatorSignedParameters :variable*8;
 uint8_t signature[remainder()];
} :chunkLength*8;
```

initiatorSessionID : The Session ID to be used by the responder when sending packets to the Initiator.



cookieLength : VLU, the length of the following cookieEcho field in bytes.

cookieEcho : The cookie from the Responder Hello, unaltered.

certLength : VLU, the length of the following initiatorCertificate field in bytes.

initiatorCertificate : The initiator's identity credentials.

skicLength : VLU, the length of the following sessionKeyInitiatorComponent field in bytes.

sessionKeyInitiatorComponent : The initiator's portion of the session key negotiation according to the Cryptography Profile.

initiatorSignedParameters : The payload portion of this chunk up to the signature field.

signature : The initiator's digital signature of the initiatorSignedParameters according to the Cryptography Profile.

Note: this specification doesn't mandate a specific choice of cryptography. The Cryptography Profile determines the syntax, algorithms, and interpretation of the initiatorCertificate, responderCertificate, sessionKeyInitiatorComponent, sessionKeyResponderComponent, and signature, and how the sessionKeyInitiatorComponent and sessionKeyResponderComponent are combined to derive the session keys.

The use of IIKeying is detailed in Section 3.5.1.

#### 2.3.8. Responder Initial Keying Chunk (RIKeying)

This chunk is sent by a responder in response to an Initiator Initial Keying as the final phase of session startup. This chunk is only allowed in a packet encrypted with the default session key, having packet mode 3 (Startup), and sent to the initiator with the Session ID specified by the initiatorSessionID field from the Initiator Initial Keying.

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x78 | chunkLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| responderSessionID |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+-----+-----+-----+-----+-----+-----+-----+-----+
| skrcLength \ | sessionKeyResponderComponent |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| signature |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```
struct rikeyingChunkPayload_t
```

```
{
 struct
 {
 uint32_t responderSessionID;
 vlu_t skrcLength :variable*8;
 uint8_t sessionKeyResponderComponent[skrcLength];
 } responderSignedParametersPortion :variable*8;
 uint8_t signature[remainder()];
} :chunkLength*8;
```

```
struct
{
 responderSignedParametersPortion;
 sessionKeyInitiatorComponent;
} responderSignedParameters;
```

responderSessionID : The Session ID to be used by the Initiator when sending packets to the Responder.

skrcLength : VLU, the length of the following sessionKeyResponderComponent field in bytes.

sessionKeyResponderComponent : The responder's portion of the session key negotiation according to the Cryptography Profile.

responderSignedParametersPortion : The payload portion of this chunk up to the signature field.

signature : The responder's digital signature of the responderSignedParameters (see below) according to the Cryptography Profile.

`responderSignedParameters` : The concatenation of the  
     `responderSignedParametersPortion` (the payload portion of this  
     chunk up to the signature field) and the  
     `sessionKeyInitiatorComponent` from the Initiator Initial Keying to  
     which this chunk is a response.

Note: this specification doesn't mandate a specific choice of cryptography. The Cryptography Profile determines the syntax, algorithms, and interpretation of the `initiatorCertificate`, `responderCertificate`, `sessionKeyInitiatorComponent`, `sessionKeyResponderComponent`, and signature, and how the `sessionKeyInitiatorComponent` and `sessionKeyResponderComponent` are combined to derive the session keys.

Once the responder has computed the `sessionKeyResponderComponent`, it has all of the information and state necessary for an established session with the initiator. Once the responder has sent this chunk to the initiator, the session is established and ready to carry flows of user data.

Once the initiator receives, verifies, and processes this chunk, it has all of the information and state necessary for an established session with the responder. The session is established and ready to carry flows of user data.

The use of RIKeying is detailed in Section 3.5.1.

### 2.3.9. Ping Chunk

This chunk is sent in order to elicit a Ping Reply from the receiver. It is only allowed in a packet belonging to an established session and having packet mode 1 or 2.

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+-----+-----+-----+-----+-----+-----+
| 0x01 | chunkLength |
+-----+-----+-----+-----+-----+-----+
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+
| message |
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+

```

```

struct pingChunkPayload_t
{
 uint8_t message[chunkLength];
} :chunkLength*8;

```

message : The (potentially empty) message that is expected to be returned by the other end of the session in a Ping Reply.

The receiver of this chunk SHOULD reply as immediately as is practical with a Ping Reply.

Ping and the expected Ping Reply are typically used for session keepalive, endpoint address change verification, and path MTU discovery. See Section 3.5.4 for details.

#### 2.3.10. Ping Reply Chunk

This chunk is sent in response to a Ping chunk. It is only allowed in a packet belonging to an established session and having packet mode 1 or 2.

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x41 | chunkLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+~::~+
| messageEcho
+~::~/

```

```

struct pingReplyChunkPayload_t
{
 uint8_t messageEcho[chunkLength];
} :chunkLength*8;

```

messageEcho : The message from the Ping to which this is a response, unaltered.

#### 2.3.11. User Data Chunk

This chunk is the basic unit of transmission for the user messages of a flow. A user message comprises one or more fragments. Each fragment is carried in its own chunk and has a unique sequence number in its flow. It is only allowed in a packet belonging to an established session and having packet mode 1 or 2.

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x10 | chunkLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+
O	r	F	r	A	F
P	s	R	s	B	I
T	v	A	v	N	N
+---+---+---+---+---+					
+-----/-+-----/-+					
flowID \	seq# \	fsnOffset \			
+-----/-+-----/-+					
+~~~/~~~/~~~~~+ +~~~/~~~/~~~~~+-----/-+					
L \ T \ V	... options ...	L \ T \ V	0 \		
\~~~/~~~/~~~~~+ [if(OPT)] +~~~/~~~/~~~~~+-----/-/					
+~~~~~+					
userData					
+~~~~~+

```

```

struct userDataChunkPayload_t
{
 bool_t optionsPresent :1; // "OPT"
 uintn_t reserved1 :1; // "rsv"
 uintn_t fragmentControl :2; // "FRA"
 // 0=whole, 1=begin, 2=end, 3=middle
 uintn_t reserved2 :2; // "rsv"
 bool_t abandon :1; // "ABN"
 bool_t final :1; // "FIN"
 vlu_t flowID :variable*8;
 vlu_t sequenceNumber :variable*8; // "seq#"
 vlu_t fsnOffset :variable*8;
 forwardSequenceNumber = sequenceNumber - fsnOffset;
 if(optionsPresent)
 optionList_t options :variable*8;
 uint8_t userData[remainder()];
} :chunkLength*8;

```

optionsPresent : If set, indicates the presence of an option list before the user data. If clear, there is no option list in this chunk.

fragmentControl : Indicates how this fragment is assembled, potentially with others, into a complete user message. Possible values:

- 0 : This fragment is a complete message.
- 1 : This fragment is the first of a multi-fragment message.
- 2 : This fragment is the last of a multi-fragment message.
- 3 : This fragment is in the middle of a multi-fragment message.

A single-fragment user message has a fragment control of "0-whole". When a message has more than one fragment, the first fragment has a fragment control of "1-begin", then zero or more "3-middle" fragments, and finally a "2-end" fragment. The sequence numbers of a multi-fragment message MUST be contiguous.

abandon : If set, this sequence number has been abandoned by the sender. The userData, if any, MUST be ignored.

final : If set, this is the last sequence number of the flow.

flowID : VLU, the flow identifier.

sequenceNumber : VLU, the sequence number of this fragment.  
Fragments are assigned contiguous increasing sequence numbers in a flow. The first sequence number of a flow SHOULD be 1. The first sequence number of a flow MUST be greater than zero. Sequence numbers are unbounded and do not wrap.

fsnOffset : VLU, the difference between the Sequence Number and the Forward Sequence Number. This field MUST NOT be zero if the abandon flag is not set. This field MUST NOT be greater than sequenceNumber.

forwardSequenceNumber : The flow sender will not send (or resend) any fragment with a sequence number less than or equal to the forward sequence number.

options : If the optionsPresent flag is set, a list of zero or more Options terminated by a Marker is present. See Section 2.3.11.1 for defined options.

userData : The actual user data for this fragment.

The use of User Data is detailed in Section 3.6.2.

#### 2.3.11.1. Options for User Data

This section lists options that may appear in User Data option lists. A conforming implementation MUST support the options in this section.

A flow receiver MUST reject a flow containing a flow option that is not understood if the option type is less than 8192 (0x2000). A flow receiver MUST ignore any flow option that is not understood if the option type is 8192 or greater.

The following option type codes are defined for User Data:

0x00 : Users's Per-Flow Metadata (Section 2.3.11.1.1)

0x0a : Return Flow Association (Section 2.3.11.1.2)

#### 2.3.11.1.1. User's Per-Flow Metadata

This option conveys the user's per-flow metadata for the flow to which it's attached.

```
+-----/--+-----/--+~~~~~+
| length \ | 0x00 \ | userMetadata |
+-----/--+-----/--+~~~~~/
```

```
struct userMetadataOptionValue_t
{
 uint8_t userMetadata[remainder()];
} :remainder()*8;
```

The user associates application-defined metadata with each flow. The metadata does not change over the life of the flow. Every flow MUST have metadata. A flow sender MUST send this option with the first User Data chunk for this flow in each packet until an acknowledgement for this flow is received. A flow sender SHOULD NOT send this option more than once for each flow in any one packet. A flow sender SHOULD NOT send this option for a flow once the flow has been acknowledged.

This specification doesn't mandate the encoding, syntax, or interpretation of the user's per-flow metadata; this is determined by the application.

The userMetadata SHOULD NOT exceed 512 bytes. The userMetadata MAY be 0 bytes in length.

#### 2.3.11.1.2. Return Flow Association

A new flow can be considered to be in return (or response) to a flow sent by the other endpoint. This option encodes the receive flow identifier to which this new sending flow is a response.

```

+-----/--+-----/--+-----/--+
| length \ | 0x0a \ | flowID \ |
+-----/--+-----/--+-----/--+

```

```

struct returnFlowAssociationOptionValue_t
{
 vlu_t flowID :variable*8;
} :variable*8;

```

Consider endpoints A and B. Endpoint A begins a flow with identifier 5 to endpoint B. A is the flow sender for A's flowID=5 and B is the flow receiver for A's flowID=5. B begins a return flow with identifier 7 to A in response to A's flowID=5. B is the flow sender for B's flowID=7 and A is the flow receiver for B's flowID=7. B sends this option with flowID set to 5 to indicate that B's flowID=7 is in response to and associated with A's flowID=5.

If there is a return association, the flow sender **MUST** send this option with the first User Data chunk for this flow in each packet until an acknowledgement for this flow is received. A flow sender **SHOULD NOT** send this option more than once for each flow in any one packet. A flow sender **SHOULD NOT** send this option for a flow once the flow has been acknowledged.

A flow **MUST NOT** indicate more than one return association.

A flow **MUST** indicate its return association, if any, upon its first transmission of a User Data chunk. A return association can't be added to a sending flow after it begins.

A flow receiver **MUST** reject a new receiving flow having a return flow association that does not indicate an F\_OPEN sending flow.

#### 2.3.12. Next User Data Chunk

This chunk is equivalent to the User Data Chunk for purposes of sending the user messages of a flow. When used, it **MUST** follow a User Data or another Next User Data chunk in the same packet.



```

0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x11 | chunkLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+
O	r	F	r	A	F
P	s	R	s	B	I
T	v	A	v	N	N
+---+---+---+---+---+---+					
+~~~/~~~/~~~/~~~/~~~/~~~/+-----+---+					
L \ T \ V	... options ...	L \ T \ V	0 \		
\~~~/~~~/~~~/~~~/~~~/~~~/+ [if(OPT)] +~~~/~~~/~~~/~~~/~~~/+-----+---+					
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+					
~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+					
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+

```

```

struct nextUserDataChunkPayload_t
{
    bool_t optionsPresent :1; // "OPT"
    uintn_t reserved1 :1; // "rsv"
    uintn_t fragmentControl :2; // "FRA"
        // 0=whole, 1=begin, 2=end, 3=middle
    uintn_t reserved2 :2; // "rsv"
    bool_t abandon :1; // "ABN"
    bool_t final :1; // "FIN"
    if(optionsPresent)
        optionList_t options :variable*8;
    uint8_t userData[remainder()];
} :chunkLength*8;

```

This chunk is considered to be for the same flowID as the most recently preceding User Data or Next User Data chunk in the same packet, having the same Forward Sequence Number, and having the next sequence number. The optionsPresent, fragmentControl, abandon, and final flags, and the options (if present) have the same interpretation as for the User Data chunk.

```

...
-----+-----
10 00 07 | User Data chunk, length=7
00       | OPT=0, FRA=0 "whole", ABN=0, FIN=0
02 05 03 | flowID=2, seq#=5, fsn=(5-3)=2
00 01 02 | data 3 bytes: 00, 01, 02
-----+-----
11 00 04 | Next User Data chunk,length=4
00       | OPT=0, FRA=0 "whole", ABN=0, FIN=0
03 04 05 | flowID=2, seq#=6, fsn=2
        | data 3 bytes: 03, 04, 05
-----+-----
11 00 04 | Next User Data chunk, length=4
00       | OPT=0, FRA=0 "whole", ABN=0, FIN=0
06 07 08 | flowID=2, seq#=7, fsn=2
        | data 3 bytes: 06, 07, 08
-----+-----

```

Figure 3: Sequential messages in one packet using Next User Data

The use of Next User Data is detailed in Section 3.6.2.3.2.

#### 2.3.13. Data Acknowledgement Bitmap Chunk (Bitmap Ack)

This chunk is sent by the flow receiver to indicate to the flow sender the User Data fragment sequence numbers that have been received for one flow. It is only allowed in a packet belonging to an established session and having packet mode 1 or 2.

The flow receiver can choose to acknowledge User Data with this chunk or with a Range Ack. It SHOULD choose whichever format has the most compact encoding of the sequence numbers received.

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          0x50          |          chunkLength          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+-----/-----/-----/-----/-----/-----/-----/-----+
|   flowID   \ |   bufAvail   \ |   cumAck   \ |
+-----/-----/-----/-----/-----/-----/-----/-----+
+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C		
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		
9	8	7	6	5	4	3	2	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1		
+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+~+

```

```

struct dataAckBitmapChunkPayload_t
{
    vlu_t flowID :variable*8;
    vlu_t bufferBlocksAvailable :variable*8; // "bufAvail"
    vlu_t cumulativeAck :variable*8; // "cumAck"
    bufferBytesAvailable = bufferBlocksAvailable * 1024;
    acknowledge(0 through cumulativeAck);
    ackCursor = cumulativeAck + 1;
    while(remainder() > 0)
    {
        for(bitPosition = 8; bitPosition > 0; bitPosition--)
        {
            bool_t bit :1;
            if(bit)
                acknowledge(ackCursor + bitPosition);
        }
        ackCursor += 8;
    }
} :chunkLength*8;

```

flowID : VLU, the flow identifier.

bufferBlocksAvailable : VLU, the number of 1024-byte blocks of User Data that the receiver is currently able to accept. Section 3.6.3.5 describes how to calculate this value.

cumulativeAck : VLU, the acknowledgement of every fragment sequence number in this flow that is less than or equal to this value. This MUST NOT be less than the highest forward sequence number received in this flow.

bit field : A sequence of zero or more bytes representing a bit field of received fragment sequence numbers after the cumulative acknowledgement, least significant bit first. A set bit indicates receipt of a sequence number. A clear bit indicates that sequence number was not received. The least significant bit of the first byte is the second sequence number following the cumulative acknowledgement, the next bit is the third sequence number following, and so on.

```

50 00 05 | Bitmap Ack, length=5 bytes
05 7f 10 | flowID=5, bufAvail=127*1024 bytes, cumAck=0..16
79 06    | 01111001 00000110 = 18, 21, 22, 23, 24, 27, 28

```

Example bitmap ack indicating acknowledgement of fragment sequence numbers 0 through 16, 18, 21 through 24, 27 and 28.

Figure 4

#### 2.3.14. Data Acknowledgement Ranges Chunk (Range Ack)

This chunk is sent by the flow receiver to indicate to the flow sender the User Data fragment sequence numbers that have been received for one flow. It is only allowed in a packet belonging to an established session and having packet mode 1 or 2.

The flow receiver can choose to acknowledge User Data with this chunk or with a Bitmap Ack. It SHOULD choose whichever format has the most compact encoding of the sequence numbers received.

```

      0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      0x51      |      chunkLength      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+-----+-----+-----+-----+-----+-----+-----+-----+
|  flowID  \ |  bufAvail  \ |  cumAck  \ |
+-----+-----+-----+-----+-----+-----+-----+-----+
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+
|  #holes-1 \ |  #recv-1  \ |
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+
:
:
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+
|  #holes-1 \ |  #recv-1  \ |
+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+~~~~~+

```

```

struct dataAckRangesChunkPayload_t
{
    vlu_t flowID :variable*8;
    vlu_t bufferBlocksAvailable :variable*8; // "bufAvail"
    vlu_t cumulativeAck :variable*8; // "cumAck"
    bufferBytesAvailable = bufferBlocksAvailable * 1024;
    acknowledge(0 through cumulativeAck);
    ackCursor = cumulativeAck;
    while(remainder() > 0)
    {
        vlu_t holesMinusOne :variable*8; // "#holes-1"
        vlu_t receivedMinusOne :variable*8; // "#recv-1"

        ackCursor++;
        rangeFrom = ackCursor + holesMinusOne + 1;
        rangeTo = rangeFrom + receivedMinusOne;
        acknowledge(rangeFrom through rangeTo);

        ackCursor = rangeTo;
    }
} :chunkLength*8;

```

flowID : VLU, the flow identifier.

bufferBlocksAvailable : VLU, the number of 1024-byte blocks of User Data that the receiver is currently able to accept. Section 3.6.3.5 describes how to calculate this value.

cumulativeAck : VLU, the acknowledgement of every fragment sequence number in this flow that is less than or equal to this value. This MUST NOT be less than the highest forward sequence number received in this flow.

holesMinusOne / receivedMinusOne : Zero or more acknowledgement ranges, run-length encoded. Runs are encoded as zero or more pairs of VLU's indicating the number (minus one) of missing sequence numbers followed by the number (minus one) of received sequence numbers, starting at the cumulative acknowledgement. NOTE: If a parser syntax error is encountered here (that is, if the chunk is truncated such that not enough bytes remain to completely encode both VLU's of the acknowledgement range), then treat and process this chunk as though it was properly formed up to the last completely encoded range.

```

51 00 07 | Range Ack, length=7
05 7f 10 | flowID=5, bufAvail=127*1024 bytes, cumAck=0..16
00 00    | holes=1, received=1 -- missing 17, received 18
01 03    | holes=2, received=4 -- missing 19..20, received 21..24

```

Example range ack indicating acknowledgement of fragment sequence numbers 0 through 16, 18, 21, 22, 23, 24.

Figure 5

```

51 00 07 | Range Ack, length=9
05 7f 10 | flowID=5, bufAvail=127*1024 bytes, cumAck=0..16
00 00    | holes=1, received=1 -- missing 17, received 18
01 83    | holes=2, received=VLU parse error, ignore this range

```

Example range ack indicating acknowledgement of fragment sequence numbers 0 through 16 and 18, with a truncated last range. Note the truncation and parse error does not abort the entire chunk in this case.

Figure 6

#### 2.3.15. Buffer Probe Chunk

This chunk is sent by the flow sender in order to request the current available receive buffer (in the form of a Data Acknowledgement) for a flow. It is only allowed in a packet belonging to an established session and having packet mode 1 or 2.

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      0x18      |      chunkLength      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+-----/-+
|  flowID      \ |
+-----/-+

```

```

struct bufferProbeChunkPayload_t
{
    vlu_t flowID :variable*8;
} :chunkLength*8;

```

flowID : VLU, the flow identifier.

The receiver of this chunk SHOULD reply as immediately as is practical with a Data Acknowledgement.

#### 2.3.16. Flow Exception Report Chunk

This chunk is sent by the flow receiver to indicate that it is not (or is no longer) interested in the flow and would like the flow sender to close the flow. This chunk SHOULD precede every Data Acknowledgement chunk for the same flow in this condition.

This chunk is only allowed in a packet belonging to an established session and having packet mode 1 or 2.

```

 0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      0x5e      |      chunkLength      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+-----/-+-----/-+
|  flowID      \ | exception \ |
+-----/-+-----/-+

```

```

struct flowExceptionReportChunkPayload_t
{
    vlu_t flowID :variable*8;
    vlu_t exception :variable*8;
} :chunkLength*8;

```

flowID : VLU, the flow identifier.

exception : VLU, the application-defined exception code being reported.

A receiving RTMFP might reject a flow automatically, for example if

it is missing metadata, or if an invalid return association is specified. In circumstances where an RTMFP rejects a flow automatically, the exception code **MUST** be 0. The application can specify any exception code, including 0, when rejecting a flow. All non-zero exception codes are reserved for the application.

#### 2.3.17. Session Close Request Chunk (Close)

This chunk is sent to cleanly terminate a session. It is only allowed in a packet belonging to an established or closing session and having packet mode 1 or 2.

```

  0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           0x0c           |           0           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

This chunk has no payload.

The use of Close is detailed in Section 3.5.5.

#### 2.3.18. Session Close Acknowledgement Chunk (Close Ack)

This chunk is sent in response to a Session Close Request to indicate the sender has terminated the session. It is only allowed in a packet belonging to an established or closing session and having packet mode 1 or 2.

```

  0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           0x4c           |           0           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

This chunk has no payload.

The use of Close Ack is detailed in Section 3.5.5.

### 3. Operation



## 3.1. Overview

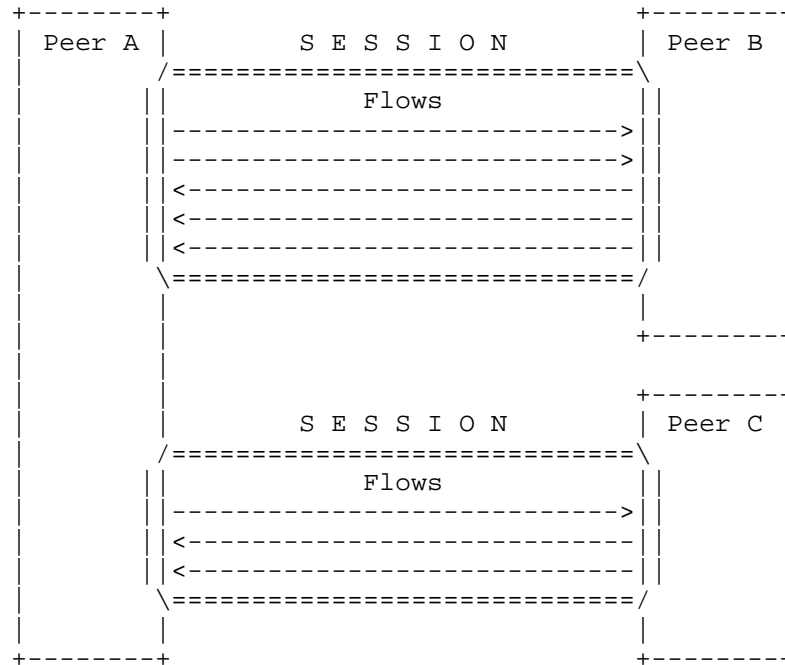


Figure 7: Sessions between pairs of communicating endpoints

Between any pair of communicating endpoints is a single, bidirectional, secured, congestion controlled session. Unidirectional flows convey messages from one end to the other within the session.

An endpoint initiates a session to a far end when communication is desired. An initiator begins with one or more candidate destination socket addresses, and may learn and try more candidate addresses during startup handshaking. Eventually a first suitable response is received, and that endpoint is selected. Startup proceeds to the selected endpoint. In the case of session startup glare, one endpoint is the prevailing initiator and the other assumes the role of responder. Encryption keys and session identifiers are negotiated between the endpoints, and the session is established.

Each endpoint may begin sending message flows to the other end. For each flow, the far end may accept it and deliver its messages to the user, or it may reject the flow and transmit an exception to the sender. The flow receiver may close and reject a flow at a later time, after first accepting it. The flow receiver acknowledges all

data sent to it regardless of whether the flow was accepted. Acknowledgements drive a congestion control mechanism.

An endpoint may have concurrent sessions with other far endpoints. The multiple sessions are distinguished by a session identifier rather than by socket address. This allows an endpoint's address to change mid-session without having to tear down and re-establish a session. The existing cryptographic state for a session can be used to verify a change of address while protecting against session hijacking or denial-of-service.

A sender may indicate to a receiver that some user messages are of a time-critical or real-time nature. A receiver may indicate to senders on concurrent sessions that it is receiving time-critical messages from another endpoint. The other senders SHOULD modify their congestion control parameters to yield capacity to the session carrying time-critical messages.

A sender may close a flow. The flow is completed when the receiver has no outstanding gaps before the final fragment of the flow. The sender and receiver reserve a completed flow's identifier for a time to allow in flight messages to drain from the network.

Eventually, neither end will have any flows open to the other. The session will be idle and quiescent. Either end may reliably close the session to recover its resources.

In certain circumstances, an endpoint may be ceasing operation and not have time to wait for acknowledgement of a reliable session close. In this case the halting endpoint may send an abrupt session close to advise the far end that it is halting immediately.

### 3.2. Endpoint Identity

Each RTMFP endpoint has an identity. The identity is encoded in a certificate. This specification doesn't mandate any particular certificate format, cryptographic algorithms, or cryptographic properties for certificates.

An endpoint is named by an Endpoint Discriminator. This specification doesn't mandate any particular format for Endpoint Discriminators.

An Endpoint Discriminator MAY select more than one identity, and MAY match more than one distinct certificate.

Multiple distinct Endpoint Discriminators MAY match one certificate.

It is RECOMMENDED that multiple endpoints not have the same identity. Entities with the same identity are indistinguishable during session startup, which could be undesirable in some applications.

An endpoint MAY have more than one address.

The Cryptography Profile implements the following functions for identities, certificates, and Endpoint Discriminators, whose operation MUST be deterministic:

- o Test whether a given certificate is authentic. Authenticity can comprise verifying an issuer signature chain in a public key infrastructure.
- o Test whether a given Endpoint Discriminator selects a given certificate.
- o Test whether a given Endpoint Discriminator selects the local endpoint.
- o Generate a Canonical Endpoint Discriminator for a given certificate. Canonical Endpoint Discriminators for distinct identities SHOULD be distinct. If two distinct identities have the same Canonical Endpoint Discriminator, an initiator might abort a new opening session to the second identity (Section 3.5.1.1.1), which might not be desired.
- o Given a certificate, a message, and a digital signature over the message, test whether the signature is valid and generated by the owner of the certificate.
- o Generate a digital signature for a given message corresponding to the near identity.
- o Given the near identity and a far certificate, determine which one shall prevail as Initiator and which shall assume the Responder role in the case of startup glare. The far end MUST arrive at the same conclusion. A comparison function can comprise performing a lexicographic ordering of the binary certificates, and declaring the far identity the prevailing endpoint if the far certificate is ordered before the near certificate, and otherwise declaring the near identity to be the prevailing endpoint.
- o Given a first certificate and a second certificate, test whether a new incoming session from the second shall override an existing session with the first. It is RECOMMENDED that the test comprise testing whether the certificates are bitwise identical.

All other semantics for certificates and Endpoint Discriminators are determined by the Cryptography Profile and the application.

### 3.3. Packet Multiplex

An RTMFP typically has one or more interfaces through which it communicates with other RTMFP endpoints. RTMFP can communicate with multiple distinct other RTMFP endpoints through each local interface. Session multiplexing over a shared interface can facilitate peer-to-peer communications through a NAT, by enabling third party endpoints such as Forwarders (Section 3.5.1.5) and Redirectors (Section 3.5.1.4) to observe the translated public address and inform peers of the translation.

An interface is typically a UDP socket (Section 2.2.1), but MAY be any suitable datagram transport service where endpoints can be addressed by IPv4 or IPv6 socket addresses.

RTMFP uses a session ID to multiplex and demultiplex communications with distinct endpoints (Section 2.2.2), in addition to the endpoint socket address. This allows an RTMFP to detect a far-end address change (as might happen for example in mobile and wireless scenarios) and for communication sessions to survive address changes. This also allows an RTMFP to act as a Forwarder or Redirector for an endpoint with which it has an active session, by distinguishing startup packets from those of the active session.

On receiving a packet, an RTMFP decodes the session ID to look up the corresponding session information context and decryption key. Session ID 0 is reserved for session startup and MUST NOT be used for an active session. A packet for session ID 0 uses the Default Session Key as defined by the Cryptography Profile.

### 3.4. Packet Fragmentation

When an RTMFP packet (Section 2.2.4) is unavoidably larger than the path MTU (such as a startup packet containing an RHello (Section 2.3.4) or IIKeying (Section 2.3.7) chunk with a large certificate), it can be fragmented into segments that do not exceed the path MTU using the Packet Fragment chunk (Section 2.3.1).

The packet fragmentation mechanism SHOULD be used only to segment unavoidably large packets. Accordingly, this mechanism SHOULD be employed only during session startup with session ID 0. This mechanism MUST NOT be used instead of the natural fragmentation mechanism of the User Data (Section 2.3.11) and Next User Data (Section 2.3.12) chunks for dividing the messages of the user's data flows into segments that do not exceed the path MTU.

A fragmented plain RTMFP packet is reassembled by concatenating the packetFragment fields of the fragments for the packet in contiguous ascending order, starting from index 0 through and including the final fragment.

When reassembling packets for Session ID 0, a receiver SHOULD identify the packets by both the socket address from which the packet containing the fragment was received as well as the indicated packetID.

A receiver SHOULD allow up to 60 seconds to completely receive a fragmented packet for which progress is being made. A packet is progressing if at least one new fragment for it was received in the last second.

A receiver MUST discard a Packet Fragment chunk having an empty packetFragment field.

The mode of each packet containing Packet Fragments for the same fragmented packet MUST match the mode of the fragmented packet. A receiver MUST discard any new Packet Fragment chunk received in a packet with a mode different from the mode of the packet containing the first received fragment. A receiver MUST discard any reassembled packet with a mode different than the packets containing its fragments.

In order to avoid jamming the network, the sender MUST rate limit packet transmission. In the absence of specific path capacity information (for instance, during session startup), a sender SHOULD NOT send more than 4380 bytes nor more than four packets per distinct endpoint every 200ms.

To avoid resource exhaustion, a receiver SHOULD limit the number of concurrent packet reassembly buffers and the size of each buffer. Limits can depend on the expected size of reassembled packets, the rate at which fragmented packets are expected to be received and degree of interleaving, and the expected function of the receiver. Limits can depend on the available resources of the receiver. There can be different limits for packets with Session ID 0 and packets for established sessions. For example, a busy server might need to allow for several hundred concurrent packet reassembly buffers to accommodate hundreds of connection requests per second with potentially interleaved fragments, but a client device with constrained resources could allow just a few reassembly buffers. In the absence of specific information regarding the expected size of reassembled packets, a receiver should set the limit for each packet reassembly buffer to 65536 bytes.

### 3.5. Sessions

A session is the protocol relationship between a pair of communicating endpoints, comprising the shared and endpoint-specific information context necessary to carry out the communication. The session context at each end includes at least:

- o TS\_RX: the last timestamp received from the far end;
- o TS\_RX\_TIME: the time at which TS\_RX was first observed to be different than its previous value;
- o TS\_ECHO\_TX: the last timestamp echo sent to the far end;
- o MRT0: the measured retransmission timeout;
- o ERT0: the effective retransmission timeout;
- o Cryptographic keys for encrypting and decrypting packets, and for verifying the validity of packets, according to the Cryptography Profile;
- o Cryptographic near and far nonces, according to the Cryptography Profile, the near nonce being the far end's far nonce, and vice versa;
- o The certificate of the far end;
- o The receive session identifier, used by the far end when sending packets to this end;
- o The send session identifier to use when sending packets to the far end;
- o DESTADDR: the socket address to which to send packets to the far end;
- o The set of all sending flow contexts (Section 3.6.2);
- o The set of all receiving flow contexts (Section 3.6.3);
- o The transmission budget, which controls the rate at which data is sent into the network (for example, a congestion window);
- o S\_OUTSTANDING\_BYTES: the total amount of user message data outstanding, or in flight, in the network; that is the sum of the F\_OUTSTANDING\_BYTES of each sending flow in the session;

- o RX\_DATA\_PACKETS: a count of the number of received packets containing at least one User Data chunk since the last acknowledgement was sent, initially 0;
- o ACK\_NOW: a boolean flag indicating whether an acknowledgement should be sent immediately, initially false;
- o DELACK\_ALARM: an alarm to trigger an acknowledgement after a delay, initially unset;
- o The state, at any time being one of the following values: the opening states S\_IHELLO\_SENT and S\_KEYING\_SENT; the open state S\_OPEN; the closing states S\_NEARCLOSE and S\_FARCLOSE\_LINGER; and the closed states S\_CLOSED and S\_OPEN\_FAILED; and
- o The role of this end of the session, which is either Initiator or Responder.

Note: this diagram is only a summary of state transitions and their causing events, and is not a complete operational specification.

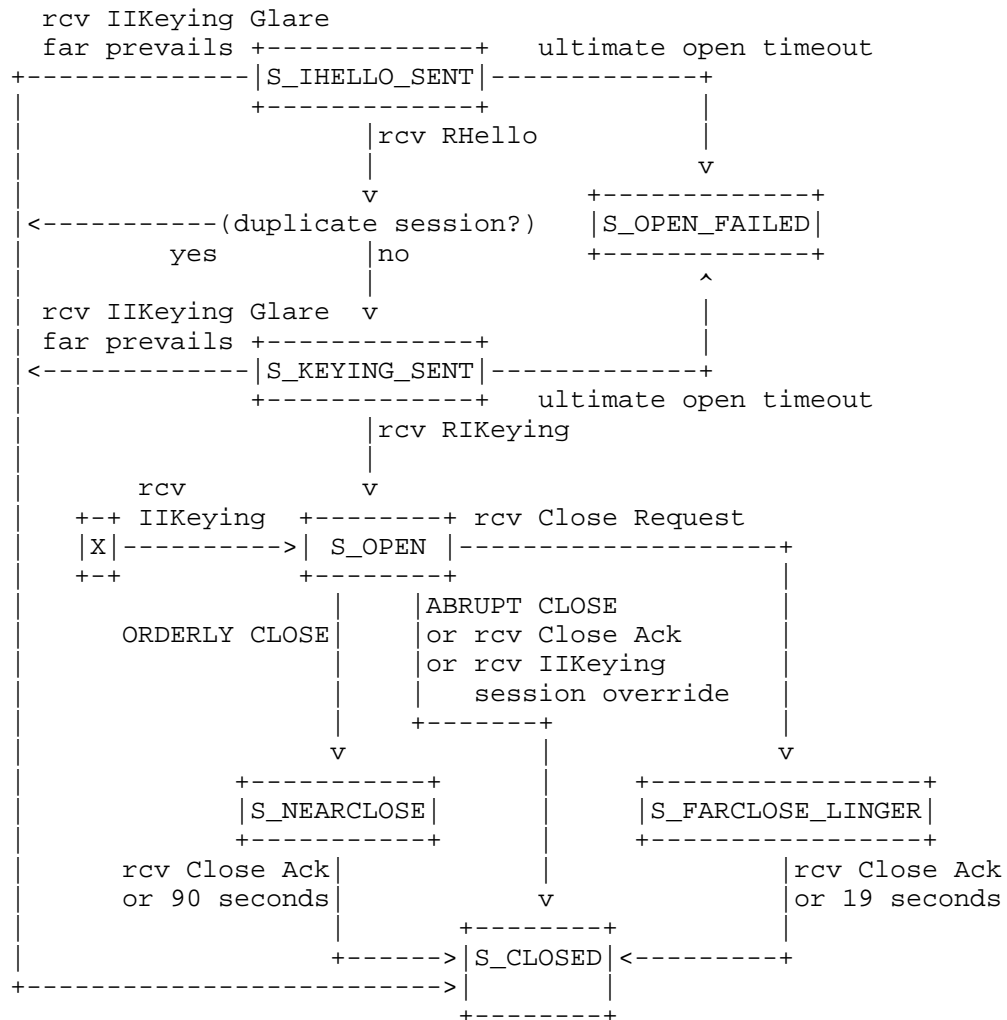


Figure 8: Session state diagram

### 3.5.1. Startup

#### 3.5.1.1. Normal Handshake

RTMFP sessions are established with a 4-way handshake in two round trips. The Initiator begins by sending an IHello to one or more candidate addresses for the desired destination endpoint. A



Responder statelessly sends an RHello in response. The first correct RHello received at the Initiator is selected; all others are ignored. The Initiator computes its half of the session keying and sends an IIKeying. The Responder receives the IIKeying, and if it is acceptable, computes its half of the session keying, at which point it can also compute the shared session keying and session nonces. The Responder creates a new S\_OPEN session with the Initiator, and sends an RIKeying. The Initiator receives the RIKeying, and if it is acceptable, it computes the shared session keying and session nonces. The Initiator's session is now S\_OPEN.

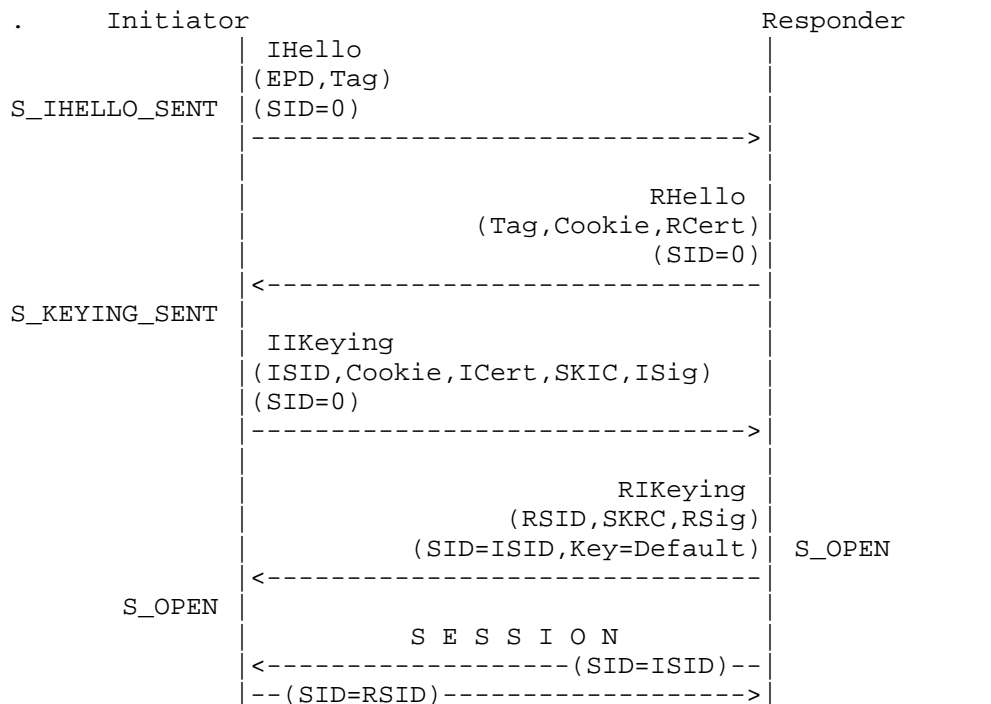


Figure 9: Normal handshake

In the following sections the handshake is detailed from the perspectives of the Initiator and Responder.

#### 3.5.1.1.1. Initiator

The Initiator determines that a session is needed for an Endpoint Discriminator. The Initiator creates state for a new opening session and begins with a candidate endpoint address set containing at least one address. The new session is placed in the S\_IHELLO\_SENT state.

If the session does not move to the S\_OPEN state before an ultimate open timeout, the session has failed and moves to the S\_OPEN\_FAILED state. The RECOMMENDED ultimate open timeout is 95 seconds.

The Initiator chooses a new, unique Tag not used by any currently opening session. It is RECOMMENDED that the Tag be cryptographically pseudorandom and be at least 8 bytes in length, so that it is hard to guess. The Initiator constructs an IHello chunk (Section 2.3.2) with the Endpoint Discriminator and the Tag.

While the Initiator is in the S\_IHELLO\_SENT state, it sends the IHello to each candidate endpoint address in the set, on a backoff schedule. The backoff SHOULD NOT be less than multiplicative with not less than 1.5 seconds added to the interval between each attempt. The backoff SHOULD be scheduled separately for each candidate address, since new candidates can be added over time.

If the Initiator receives a Redirect chunk (Section 2.3.5) with a Tag Echo matching this session, AND this session is in the S\_IHELLO\_SENT state, then for each redirect destination indicated in the Redirect: if the candidate endpoint address set contains fewer than REDIRECT\_THRESHOLD addresses, add the indicated redirect destination to the candidate endpoint address set. REDIRECT\_THRESHOLD SHOULD NOT be more than 24.

If the Initiator receives an RHello chunk (Section 2.3.4) with a Tag Echo matching this session, AND this session is in the S\_IHELLO\_SENT state, AND the Responder certificate matches the desired Endpoint Discriminator, AND the certificate is authentic according to the Cryptography Profile, then:

1. If the Canonical Endpoint Discriminator for the responder certificate matches the Canonical Endpoint Discriminator of another existing session in the S\_KEYING\_SENT or S\_OPEN states, AND the certificate of the other opening session matches the desired Endpoint Discriminator, then: this session is a duplicate and SHOULD be aborted in favor of the other existing session; otherwise
2. Move to the S\_KEYING\_SENT state. Set DESTADDR, the far end address for the session, to the address from which this RHello was received. The Initiator chooses a new, unique receive session ID, not used by any other session, for the Responder to use when sending packets to the Initiator. It computes a Session Key Initiator Component appropriate to the responder's certificate according to the Cryptography Profile. Using this data and the cookie from the RHello, the Initiator constructs and signs an IIKeying chunk (Section 2.3.7).

While the Initiator is in the S\_KEYING\_SENT state, it sends the IIKeying to DESTADDR on a backoff schedule. The backoff SHOULD NOT be less than multiplicative with not less than 1.5 seconds added to the interval between each attempt.

If the Initiator receives an RIKeying chunk (Section 2.3.8) in a packet with this session's receive session identifier, AND this session is in the S\_KEYING\_SENT state, AND the signature in the chunk is authentic according to the far end's certificate (from the RHello), AND the Session Key Responder Component successfully combines with the Session Key Initiator Component and the near and far certificates to form the shared session keys and nonces according to the Cryptography Profile, then the session has opened successfully. The session moves to the S\_OPEN state. The send session identifier is set from the RIKeying. Packet encryption, decryption, and verification now use the newly computed shared session keys, and the session nonces are available for application-layer cryptographic challenges.

#### 3.5.1.1.2. Responder

On receipt of an IHello chunk (Section 2.3.2) with an Endpoint Discriminator that selects its identity, an endpoint SHOULD construct an RHello chunk (Section 2.3.4) and send it to the address from which the IHello was received. To avoid a potential resource exhaustion denial-of-service, the endpoint SHOULD NOT create any persistent state associated with the IHello. The endpoint MUST generate the cookie for the RHello in such a way that it can be recognized as authentic and valid when echoed in an IIKeying. The endpoint SHOULD use the address from which the IHello was received as part of the cookie generation formula. Cookies SHOULD be valid only for a limited time; that lifetime SHOULD NOT be less than 95 seconds (the recommended ultimate session open timeout).

On receipt of an FIHello chunk (Section 2.3.3) from a Forwarder (Section 3.5.1.5) where the Endpoint Discriminator selects its identity, an endpoint SHOULD do one of the following:

1. Compute, construct and send an RHello as though the FIHello was an IHello received from the indicated reply address; or
2. Construct and send an Implied Redirect (Section 2.3.5) to the FIHello's reply address; or
3. Ignore this FIHello.

On receipt of an IIKeying chunk (Section 2.3.7), if the cookie is not authentic or if it has expired, ignore this IIKeying; otherwise,

On receipt of an IIKeying chunk, if the cookie appears authentic but does not match the address from which the IIKeying's packet was received, perform the special processing at Cookie Change (Section 3.5.1.2); otherwise,

On receipt of an IIKeying with an authentic and valid cookie, if the certificate is authentic according to the Cryptography Profile, AND the signature in the chunk is authentic according to the far end's certificate and the Cryptography Profile, AND the Session Key Initiator Component is acceptable, then:

1. If the address from which this IIKeying was received corresponds to an opening session in the S\_IHELLO\_SENT or S\_KEYING\_SENT state, perform the special processing at Glare (Section 3.5.1.3); otherwise,
2. If the address from which this IIKeying was received corresponds to a session in the S\_OPEN state, then:
  1. If the receiver was the Responder for the S\_OPEN session and the session identifier, certificate, and Session Key Initiator Component are identical to those of the S\_OPEN session, this IIKeying is a retransmission, so resend the S\_OPEN session's RIKeying using the Default Session Key as specified below; otherwise,
  2. If the certificate from this IIKeying does not override the certificate of the S\_OPEN session: ignore this IIKeying; otherwise,
  3. The certificate from this IIKeying overrides the certificate of the S\_OPEN session; this is a new opening session from the same identity and the existing S\_OPEN session is stale. Move the existing S\_OPEN session to S\_CLOSED and abort all of its flows (signaling exceptions to the user), then continue processing this IIKeying.

Otherwise,

3. Compute a Session Key Responder Component and choose a new, unique receive session ID not used by any other session for the Initiator to use when sending packets to the Responder. Using this data, construct and, with the Session Key Initiator Component, sign an RIKeying chunk (Section 2.3.8). Using the Session Key Initiator and Responder Components and the near and far certificates, the Responder combines and computes the shared session keys and nonces according to the Cryptography Profile. The Responder creates a new session in the S\_OPEN state, with the

far endpoint address DESTADDR taken from the source address of the packet containing the IIKeying and the send session identifier taken from the IIKeying. The Responder sends the RIKeying to the Initiator using the Default Session Key and the requested send session identifier. Packet encryption, decryption, and verification of all future packets for this session use the newly computed keys, and the session nonces are available for application-layer cryptographic challenges.

#### 3.5.1.2. Cookie Change

In some circumstances, the Responder may generate an RHello cookie for an Initiator's address that isn't the address the Initiator would use when sending packets directly to the Responder. This can happen, for example, when the Initiator has multiple local addresses, and uses one to reach a Forwarder (Section 3.5.1.5) but another to reach the Responder.

Consider the following example:

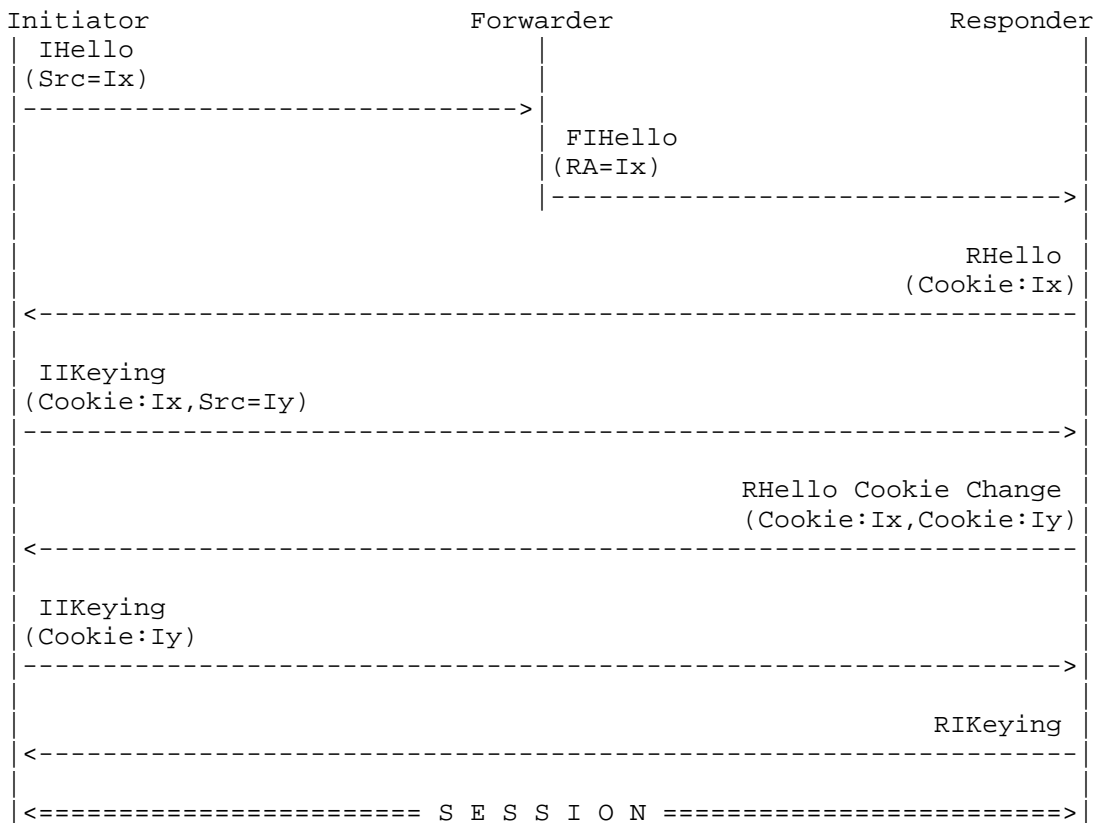


Figure 10: Handshake with Cookie Change

Initiator has two network interfaces, a first preferred interface with address `Ix = 192.0.2.100:50000`, and a second with address `Iy = 198.51.100.101:50001`. Responder has one interface with address `Ry = 198.51.100.200:51000`, on the same network as Initiator's second interface. Initiator uses its first interface to reach a Forwarder. Forwarder observes Initiator's address of `Ix` and sends a Forwarded `IHello` (Section 2.3.3) to Responder. Responder treats this as if it was an `IHello` from `Ix`, calculates a corresponding cookie, and sends an `RHello` to `Ix`. Initiator receives this `RHello` from `Ry` and selects that address as the destination for the session. It then sends an `IIKeying`, copying the cookie from the `RHello`. However, since the source of the `RHello` is `Ry`, on a network to which the Initiator is directly connected, Initiator uses its second interface `Iy` to send the `IIKeying`. Responder, on receiving the `IIKeying`, will compare the cookie to the expected value based on the source address of the packet, and since the `IIKeying` source doesn't match the `IHello` source used to generate the cookie, Responder will reject the `IIKeying`.

If Responder determines that it generated the cookie in the IIKeying but the cookie doesn't match the sender's address (for example, if the cookie is in two parts, with a first part generated independently of the Initiator's address, and a second part dependent on the address), Responder SHOULD generate a new cookie based on the address from which the IIKeying was received, and send an RHello Cookie Change chunk (Section 2.3.6) to the source of the IIKeying, using the session ID from the IIKeying and the Default Session Key.

If Initiator receives an RHello Cookie Change chunk for a session in the S\_KEYING\_SENT state, AND the old cookie matches the one originally sent to the Responder, then: Initiator adopts the new cookie, constructs and signs a new IIKeying chunk, and sends the new IIKeying to the Responder. Initiator SHOULD NOT change the cookie for a session more than once.

#### 3.5.1.3. Glare

Glare occurs when two endpoints attempt to initiate sessions to each other concurrently. Glare is detected by receipt of a valid and authentic IIKeying from an endpoint address which is a destination for an opening session. Only one session is allowed between a pair of endpoints.

Glare is resolved by comparing the certificate in the received IIKeying with the near end's certificate. The Cryptography Profile defines a certificate comparison function to determine the prevailing endpoint when there is glare.

If the near end prevails, discard and ignore the received IIKeying. The far end will abort its opening session on receipt of IIKeying from the near end.

Otherwise, the far end prevails:

1. If the certificate in the IIKeying overrides the certificate associated with the near opening session according to the Cryptography Profile, then: abort and destroy the near opening session. Then,
2. Continue with normal Responder IIKeying processing (Section 3.5.1.1.2).

## 3.5.1.4. Redirector

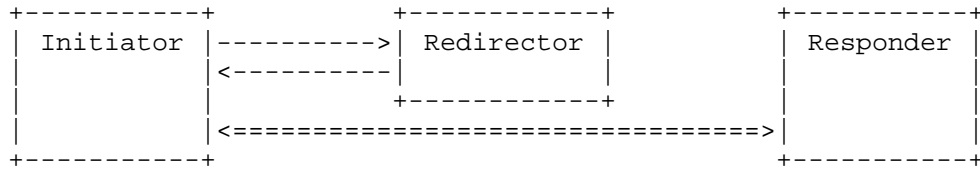


Figure 11: Redirector

A Redirector acts like a name server for Endpoint Discriminators. An Initiator MAY use a Redirector to discover additional candidate endpoint addresses for a desired endpoint.

On receipt of an IHello chunk with an Endpoint Discriminator that does not select the Redirector's identity, the Redirector constructs and sends a Responder Redirect chunk (Section 2.3.5) back to the Initiator containing one or more additional candidate addresses for the indicated endpoint.

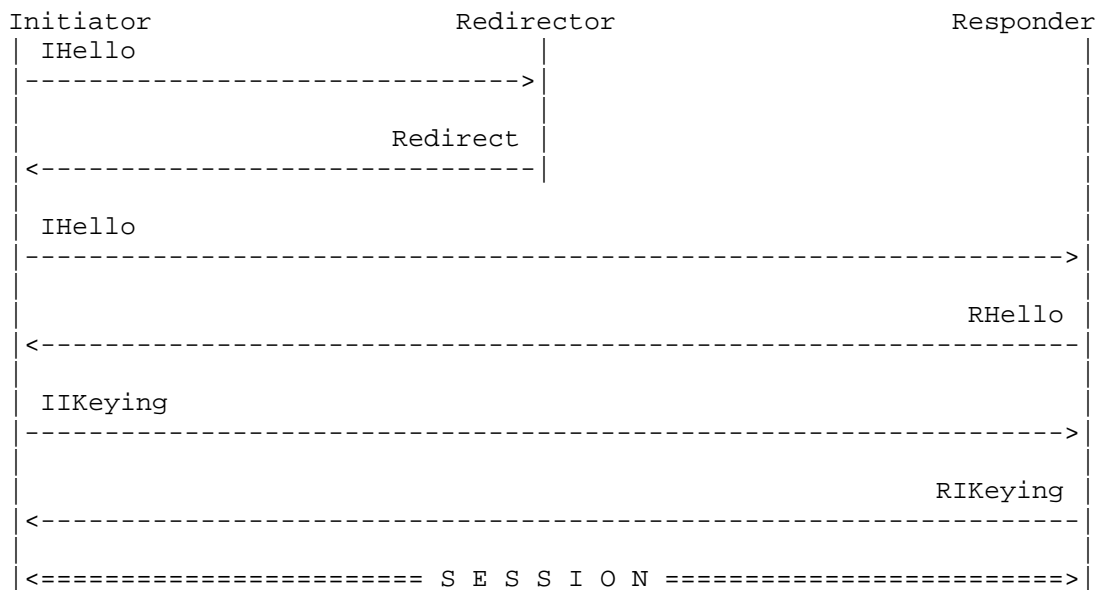


Figure 12: Handshake using a Redirector

Deployment Design Note: Redirectors SHOULD NOT initiate new sessions to endpoints that might use the Redirector's address as a candidate for another endpoint, since the far end might interpret the Redirector's IIKeying as glare for the far end's initiation to the



other endpoint.

### 3.5.1.5. Forwarder

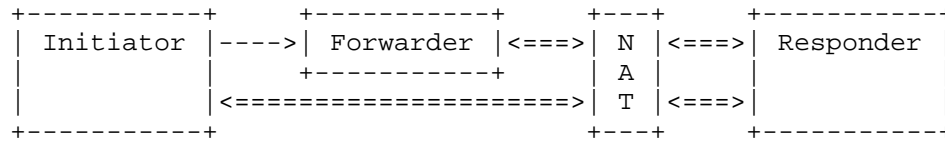


Figure 13: Forwarder

A Responder might be behind a NAT or firewall that doesn't allow inbound packets to reach the endpoint until it first sends an outbound packet for a particular far endpoint address.

A Forwarder's endpoint address MAY be a candidate address for another endpoint. A Responder MAY use a Forwarder to receive FIHello chunks sent on behalf of an Initiator.

On receipt of an IHello chunk with an Endpoint Discriminator that does not select the Forwarder's identity, if the Forwarder has an S\_OPEN session with an endpoint whose certificate matches the desired Endpoint Discriminator, the forwarder constructs and sends an FIHello chunk (Section 2.3.3) to the selected endpoint over the S\_OPEN session, using the Tag and Endpoint Discriminator from the IHello chunk and the source address of the packet containing the IHello for the corresponding fields of the FIHello.

On receipt of an FIHello chunk, a Responder might send an RHello or Implied Redirect to the original source of the IHello (Section 3.5.1.1.2), potentially allowing future packets to flow directly between the Initiator and Responder through the NAT or firewall.

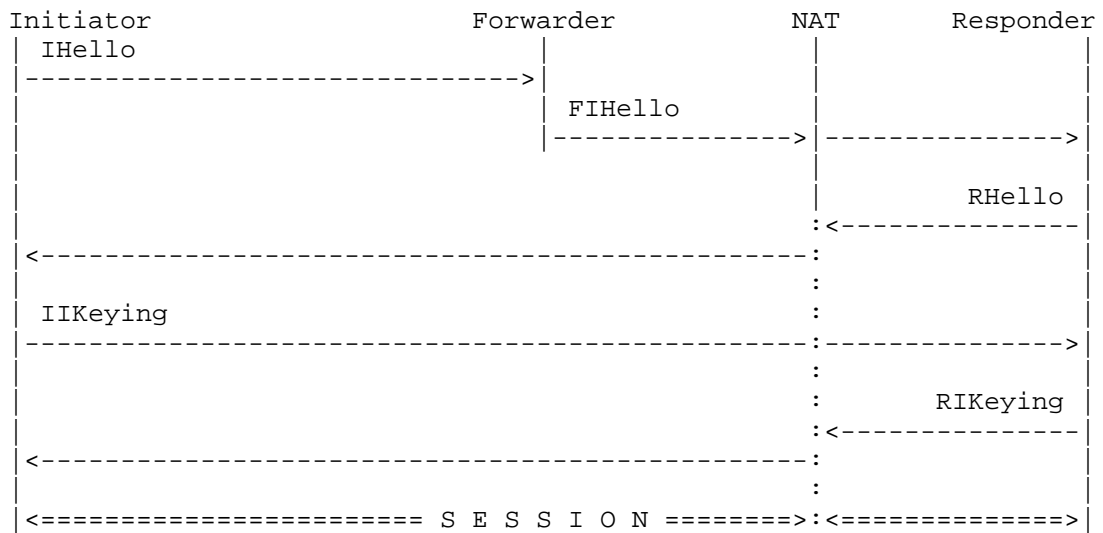


Figure 14: Forwarder handshake where Responder sends an RHello

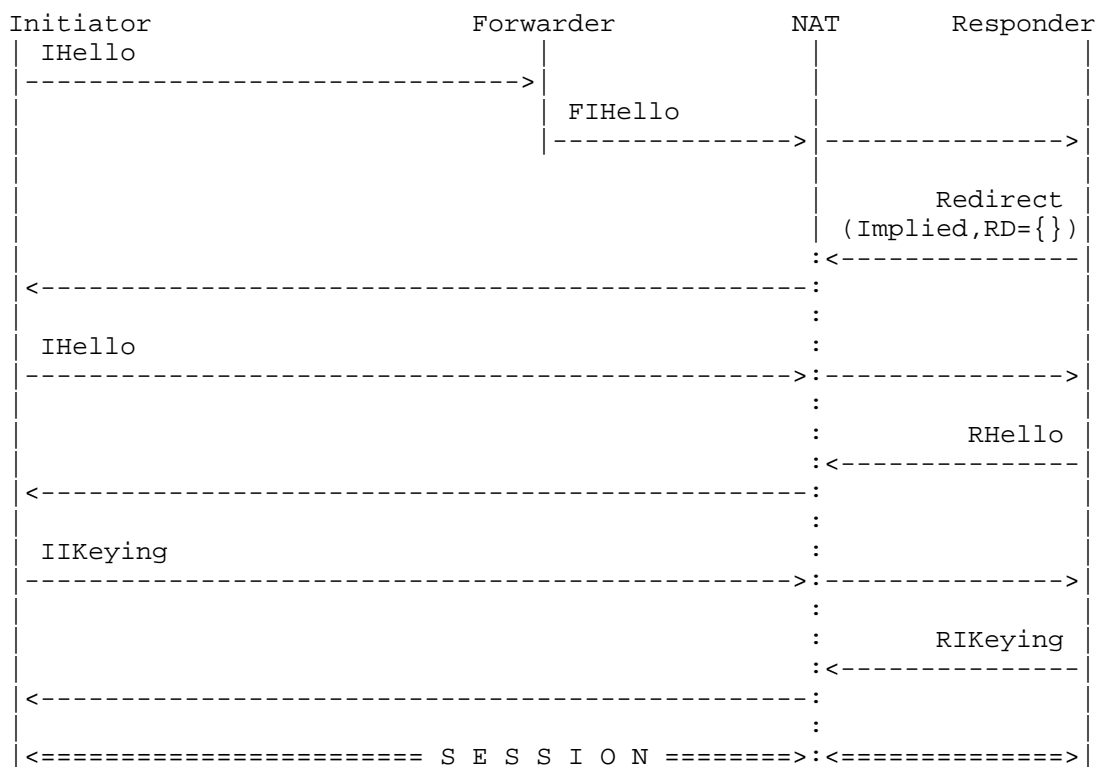


Figure 15: Forwarder handshake where Responder sends an Implied Redirect

### 3.5.1.6. Redirector and Forwarder with NAT

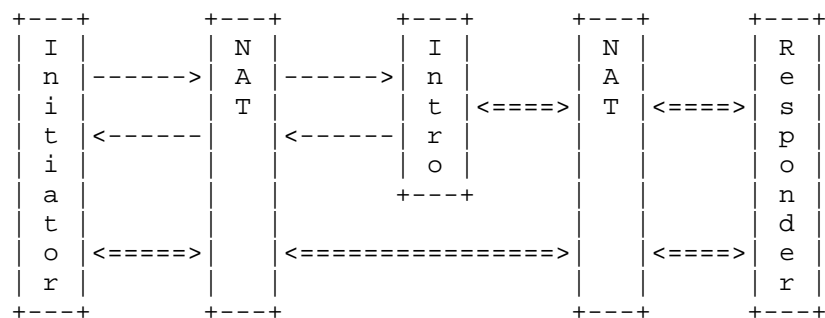


Figure 16: Introduction service for Initiator and Responder behind NATs

An Initiator and Responder might each be behind distinct NATs or

firewalls that don't allow inbound packets to reach the respective endpoints until each first sends an outbound packet for a particular far endpoint address.

An introduction service comprising Redirector and Forwarder functions may facilitate direct communication between endpoints each behind a NAT.

Responder is registered with the introduction service via an S\_OPEN session to it. The service observes and records Responder's public NAT address as the DESTADDR of the S\_OPEN session. The service MAY record other addresses for Responder, for example addresses Responder self-reports as being directly attached.

Initiator begins with an address of the introduction service as an initial candidate. The Redirector portion of the service sends a Responder Redirect to Initiator containing at least Responder's public NAT address as previously recorded. The Forwarder portion of the service sends a Forwarded IHello to Responder containing Initiator's public NAT address as observed as the source of the IHello.

Responder sends an RHello to Initiator's public NAT address in response to the FIHello. This will allow inbound packets to Responder through its NAT from Initiator's public NAT address.

Initiator sends an IHello to Responder's public NAT address in response to the Responder Redirect. This will allow inbound packets to Initiator through its NAT from Responder's public NAT address.

With transit paths created in both NATs, normal session startup can proceed.

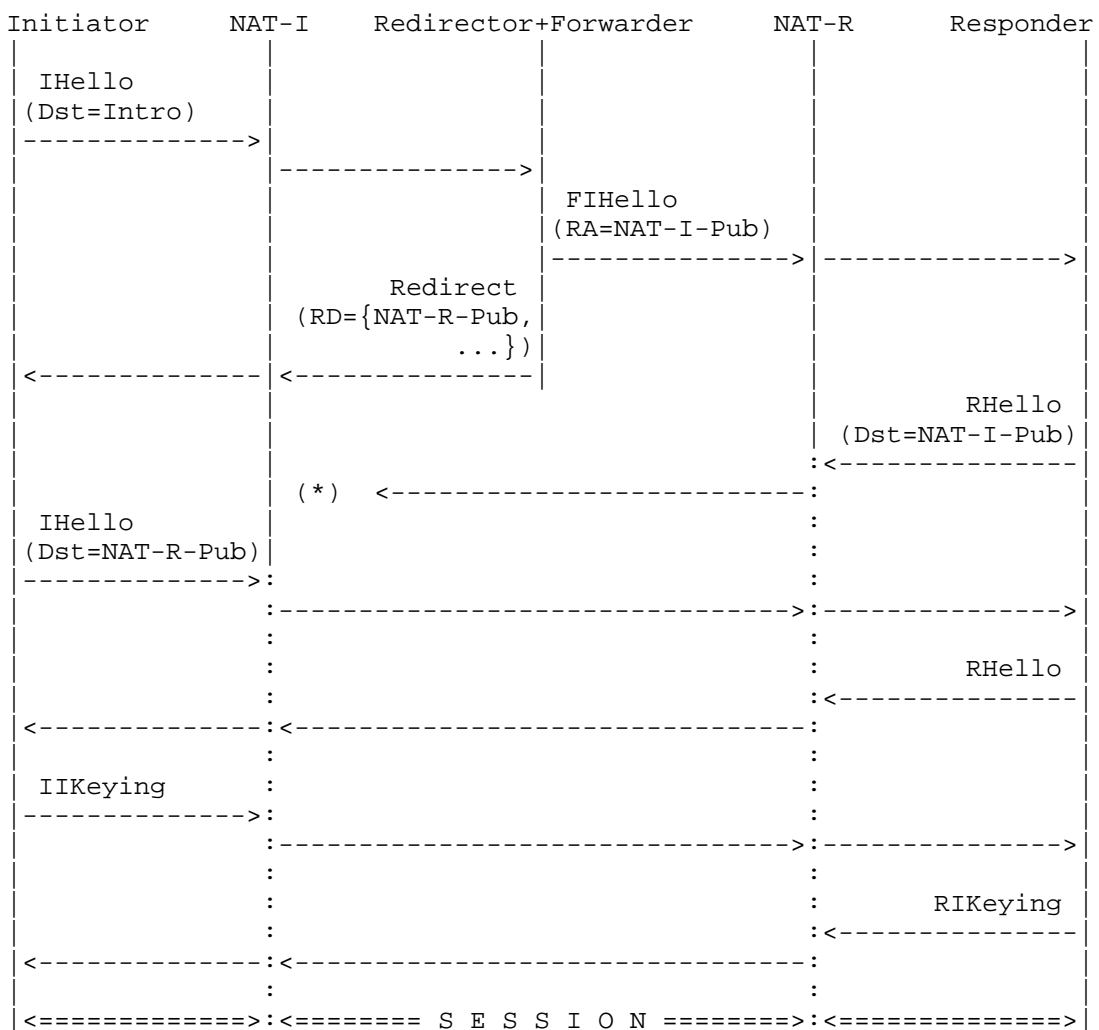


Figure 17: Handshake with Redirector and Forwarder

At the point in Figure 17 marked (\*), Responder's RHello from the FIHello might arrive at Initiator's NAT before or after Initiator's IHello is sent outbound to Responder's public NAT address. If it arrives before, it may be dropped by the NAT. If it arrives after, it will transit the NAT and trigger keying without waiting for another round trip time. The timing of this race depends, among other factors, on the relative distances of Initiator and Responder to each other and the introduction service.

## 3.5.1.1.7. Load Distribution and Fault Tolerance

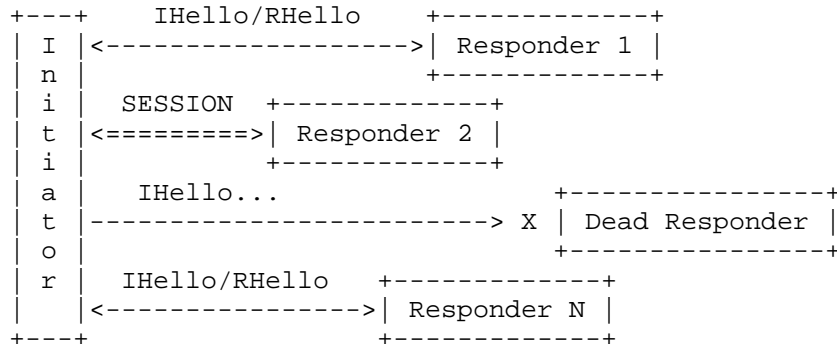


Figure 18: Parallel Open to Multiple Endpoints

Section 3.2 allows more than one endpoint to be selected by one Endpoint Discriminator. This will typically be the case for a set of servers, any of which could accommodate a connecting client.

Section 3.5.1.1.1 allows an Initiator to use multiple candidate endpoint addresses when starting a session, and specifies that the sender of the first acceptable RHello chunk to be received is selected to complete the session, with later responses ignored. An Initiator can start with the multiple candidate endpoint addresses, or it may learn them during startup from one or more Redirectors (Section 3.5.1.4).

Parallel open to multiple endpoints for the same Endpoint Discriminator combined with selection by earliest RHello can be used for load distribution and fault tolerance. The cost at each endpoint that is not selected is limited to receiving and processing an IHello, and generating and sending an RHello.

In one circumstance, multiple servers of similar processing and networking capacity may be located in near proximity to each other, such as in a data center. In this circumstance, a less heavily loaded server can respond to an IHello more quickly than more heavily loaded servers, and will tend to be selected by a client.

In another circumstance, multiple servers may be located in different physical locations, such as different data centers. In this circumstance, a server that is located nearer (in terms of network distance) to the client can respond earlier than more distant servers, and will tend to be selected by the client.

Multiple servers, in proximity or distant from one another, can form

a redundant pool of servers. A client can perform a parallel open to the multiple servers. In normal operation, the multiple servers will all respond, and the client will select one of them as described above. If one of the multiple servers fails, other servers in the pool can still respond to the client, allowing the client to successfully complete to an S\_OPEN session with one of them.

### 3.5.2. Congestion Control

An RTMFP MUST implement congestion control and avoidance algorithms that are "TCP compatible", in accordance with Internet best current practice [RFC2914]. The algorithms SHOULD NOT be more aggressive in sending data than those described in TCP Congestion Control [RFC5681], and MUST NOT be more aggressive in sending data than the "slow start algorithm" described in RFC 5681 Section 3.1.

An endpoint maintains a transmission budget in the session information context of each S\_OPEN session (Section 3.5), controlling the rate at which the endpoint sends data into the network.

For window-based congestion control and avoidance algorithms, the transmission budget is the congestion window, which is the amount of user data that is allowed to be outstanding, or in flight, in the network. Transmission is allowed when S\_OUTSTANDING\_BYTES (Section 3.5) is less than the congestion window (Section 3.6.2.3). See Appendix A for an experimental window-based congestion control algorithm for real-time and bulk data.

An endpoint avoids sending large bursts of data or packets into the network (Section 3.5.2.3).

A sending endpoint increases and decreases its transmission budget in response to acknowledgements (Section 3.6.2.4) and loss according to the congestion control and avoidance algorithms. Loss is detected by negative acknowledgement (Section 3.6.2.5) and timeout (Section 3.6.2.6).

Timeout is determined by the Effective Retransmission Timeout ERT0 (Section 3.5.2.2). ERT0 is measured using the Timestamp and Timestamp Echo packet header fields (Section 2.2.4).

A receiving endpoint acknowledges all received data (Section 3.6.3.4) to enable the sender to measure receipt of data, or lack thereof.

A receiving endpoint may be receiving time critical (or real-time) data from a first sender while receiving data from other senders. The receiving endpoint can signal its other senders (Section 2.2.4) to cause them to decrease the aggressiveness of their congestion

control and avoidance algorithms, in order to yield network capacity to the time critical data (Section 3.5.2.1).

#### 3.5.2.1. Time Critical Reverse Notification

A sender can increase its transmission budget at a rate compatible with (but not exceeding) the "slow start algorithm" specified in RFC 5681 (with which the transmission rate is doubled every round trip when beginning or restarting transmission, until loss is detected). However, a sender **MUST** behave as though the slow start threshold `SSTHRESH` is clamped to 0 (disabling the slow start algorithm's exponential increase behavior) on a session where a Time Critical Reverse Notification (Section 2.2.4) indication has been received from the far end within the last 800 milliseconds, unless the sender is itself currently sending time critical data to the far end.

During each round trip, a sender **SHOULD NOT** increase the transmission budget by more than 0.5% or by 384 bytes per round trip (whichever is greater) on a session where a Time Critical Reverse Notification indication has been received from the far end within the last 800 milliseconds, unless the sender is itself currently sending time critical data to the far end.

#### 3.5.2.2. Retransmission Timeout

RTMFP uses the Effective Retransmission Timeout `ERTO` to detect when a user data fragment has been lost in the network. The `ERTO` is typically calculated in a manner similar to that specified in Requirements for Internet Hosts - Communication Layers [RFC1122], and is a function of round trip time measurements and persistent timeout behavior.

The `ERTO` **SHOULD** be at least 250 milliseconds and **SHOULD** allow for the receiver to delay sending an acknowledgement for up to 200 milliseconds (Section 3.6.3.4.4). The `ERTO` **MUST NOT** be less than the round trip time.

To facilitate round trip time measurement, an endpoint **MUST** implement the Timestamp Echo facility:

- o On a session entering the `S_OPEN` state, initialize `TS_RX_TIME` to negative infinity, and `TS_RX` and `TS_ECHO_TX` to have no value.
- o On receipt of a packet in an `S_OPEN` session with the `timestampPresent` (Section 2.2.4) flag set, if the timestamp field in the packet is different than `TS_RX`: set `TS_RX` to the value of the timestamp field in the packet, and set `TS_RX_TIME` to the current time.



- o When sending a packet to the far end in an S\_OPEN session:
  1. Calculate `TS_RX_ELAPSED` = current time - `TS_RX_TIME`. If `TS_RX_ELAPSED` is more than 128 seconds, then set `TS_RX` and `TS_ECHO_TX` to have no value and do not include a timestamp echo; otherwise
  2. Calculate `TS_RX_ELAPSED_TICKS` to be the number of whole 4 millisecond periods in `TS_RX_ELAPSED`; then
  3. Calculate `TS_ECHO` = (`TS_RX` + `TS_RX_ELAPSED_TICKS`) MODULO 65536; then
  4. If `TS_ECHO` is not equal to `TS_ECHO_TX`, then: set `TS_ECHO_TX` to `TS_ECHO`, set the `timestampEchoPresent` flag, and set the `timestampEcho` field to `TS_ECHO_TX`.

The remainder of this section describes an OPTIONAL method for calculating the Effective Retransmission Timeout ERTO. Real-time applications and P2P mesh applications often require knowing the round trip time and RTT variance. This section additionally describes a method for measuring the round trip time and RTT variance, and calculating a smoothed round trip time.

Let the session information context contain additional variables:

- o `TS_TX`: the last timestamp sent to the far end, initialized to have no value;
- o `TS_ECHO_RX`: the last timestamp echo received from the far end, initialized to have no value;
- o `SRTT`: the smoothed round-trip time, initialized to have no value;
- o `RTTVAR`: the round-trip time variance, initialized to 0;

Initialize `MRTO` to 250 milliseconds.

Initialize `ERTO` to 3 seconds.

On sending a packet to the far end of an S\_OPEN session, if the current send timestamp is not equal to `TS_TX`, then: set `TS_TX` to the current send timestamp, set the `timestampPresent` flag in the packet header, and set the `timestamp` field to `TS_TX`.

On receipt of a packet from the far end of an S\_OPEN session, if the `timestampEchoPresent` flag is set in the packet header AND the `timestampEcho` field is not equal to `TS_ECHO_RX`, then:

1. Set TS\_ECHO\_RX to timestampEcho;
2. Calculate  $RTT\_TICKS = (current\ send\ timestamp - timestampEcho) \text{ MODULO } 65536$ ;
3. If RTT\_TICKS is greater than 32767, the measurement is invalid, so discard this measurement; otherwise
4. Calculate  $RTT = RTT\_TICKS * 4\ milliseconds$ ;
5. If SRTT has a value, then calculate new values of RTTVAR and SRTT:
  1.  $RTT\_DELTA = | SRTT - RTT |$ ;
  2.  $RTTVAR = ((3 * RTTVAR) + RTT\_DELTA) / 4$ ;
  3.  $SRTT = ((7 * SRTT) + RTT) / 8$ ;
6. If SRTT has no value, then set  $SRTT = RTT$  and  $RTTVAR = RTT / 2$ ;
7. Set  $MRTO = SRTT + 4 * RTTVAR + 200\ milliseconds$ ;
8. Set ERTO to the greater of MRTO or 250 milliseconds.

A retransmission timeout occurs when the most recently transmitted user data fragment has remained outstanding in the network for ERTO. When this timeout occurs, increase ERTO on an exponential backoff with an ultimate backoff cap of 10 seconds:

1. Calculate  $ERTO\_BACKOFF = ERTO * 1.4142$ ;
2. Calculate ERTO\_CAPPED to be the lesser of ERTO\_BACKOFF and 10 seconds;
3. Set ERTO to the greater of ERTO\_CAPPED and MRTO.

#### 3.5.2.3. Burst Avoidance

An application's sending patterns may cause the transmission budget to grow to a large value but, at times, for there to be a comparatively small amount of data outstanding in the network. In this circumstance, especially with a window-based congestion avoidance algorithm, if the application then has a large amount of new data to send (for example, a new bulk data transfer), it could send data into the network all at once to fill the window. This kind of transmission burst can jam interfaces, links, and buffers, and is undesirable.

Accordingly, in any session, an endpoint SHOULD NOT send more than six packets containing user data between receiving acknowledgements or retransmission timeouts.

The following describes an OPTIONAL method to avoid bursting large numbers of packets into the network.

Let the session information context contain an additional variable `DATA_PACKET_COUNT`, initialized to 0.

Transmission of a user data fragment on this session is not allowed if `DATA_PACKET_COUNT` is greater than or equal to 6, regardless of any other allowance of the congestion control algorithm.

On transmission of a packet containing at least one User Data chunk (Section 2.3.11), set `DATA_PACKET_COUNT = DATA_PACKET_COUNT + 1`.

On receipt of an acknowledgement chunk (Section 2.3.13, Section 2.3.14), set `DATA_PACKET_COUNT` to 0.

On a retransmission timeout, set `DATA_PACKET_COUNT` to 0.

### 3.5.3. Address Mobility

Sessions are demultiplexed with a 32 bit session ID, rather than by endpoint address. This allows an endpoint's address to change during an `S_OPEN` session. This can happen, for example, when switching from a wireless to a wired network, or when moving from one wireless base station to another, or when a NAT restarts.

If the near end receives a valid packet for an `S_OPEN` session from a source address that doesn't match `DESTADDR`, the far end might have changed addresses. The near end SHOULD verify that the far end is definitively at the new address before changing `DESTADDR`. A suggested verification method is described in Section 3.5.4.2.

### 3.5.4. Ping

If an endpoint receives a Ping chunk (Section 2.3.9) in a session in the `S_OPEN` state, it SHOULD construct and send a Ping Reply chunk (Section 2.3.10) in response if possible, copying the message unaltered. A Ping Reply response SHOULD be sent as quickly as possible following receipt of a Ping. The semantics of a Ping's message is reserved for the sender; a receiver SHOULD NOT interpret the Ping's message.

Endpoints can use the mechanism of the Ping chunk and the expected Ping Reply for any purpose. This specification doesn't mandate any

specific constraints on the format or semantics of a Ping message. A Ping Reply MUST be sent only as a response to a Ping.

Receipt of a Ping Reply implies live bidirectional connectivity. This specification doesn't mandate any other semantics for a Ping Reply.

#### 3.5.4.1. Keepalive

An endpoint can use Ping to test for live bidirectional connectivity, to test that the far end of a session is still S\_OPEN, to keep NAT translations alive, and to keep firewall holes open.

An endpoint can use Ping to hasten detection by the far end of a near end address change.

An endpoint may declare a session to be defunct and dead after a persistent failure by the far end to return Ping Replies in response to Pings.

If used for these purposes, a Keepalive Ping SHOULD have an empty message.

A Keepalive Ping SHOULD NOT be sent more often than once per ERT0. If a corresponding Ping Reply is not received within ERT0 of sending the Ping, ERT0 SHOULD be increased according to Congestion Control (Section 3.5.2).

#### 3.5.4.2. Address Mobility

This section describes an OPTIONAL but suggested method for processing and verifying a far end address change.

Let the session context contain additional variables MOB\_TX\_TS, MOB\_RX\_TS, and MOB\_SECRET. MOB\_TX\_TS and MOB\_RX\_TS have initial values of negative infinity. MOB\_SECRET should be a cryptographically pseudorandom value not less than 128 bits in length and known only to this end.

On receipt of a packet for an S\_OPEN session, after processing all chunks in the packet: if the session is still S\_OPEN, AND the source address of the packet does not match DESTADDR, AND MOB\_TX\_TS is at least one second in the past, then:

1. Set MOB\_TX\_TS to the current time;
2. Construct a Ping message comprising: a marking to indicate (to this end when returned in a Ping Reply) that it is a mobility

check (for example, the first byte being ASCII 'M' for "Mobility"), a timestamp set to MOB\_TX\_TS, and a cryptographic hash over the preceding as well as the address from which the packet was received and MOB\_SECRET; and

3. Send this Ping to the address from which the packet was received, instead of DESTADDR.

On receipt of a Ping Reply in an S\_OPEN session, if the Ping Reply's message satisfies all of these conditions:

- o it has this end's expected marking to indicate it is a mobility check, and
- o the timestamp in the message is not more than 120 seconds in the past, and
- o the timestamp in the message is greater than MOB\_RX\_TS, and
- o the cryptographic hash matches the expected value according to the contents of the message plus the source address of the packet containing this Ping Reply and MOB\_SECRET,

then:

1. Set MOB\_RX\_TS to the timestamp in the message; and
2. Set DESTADDR to the source address of the packet containing this Ping Reply.

#### 3.5.4.3. Path MTU Discovery

Packetization Layer Path MTU Discovery [RFC4821] describes a method for measuring the path MTU between communicating endpoints.

An RTMFP SHOULD perform path MTU discovery.

The method of RFC 4821 can be adapted for use in RTMFP by sending a probe packet comprising one of the Padding chunk types (type 0x00 or 0xff) and a Ping. The Ping chunk SHOULD come after the Padding chunk, to guard against a false positive response in case the probe packet is truncated.

#### 3.5.5. Close

An endpoint may close a session at any time. Typically an endpoint will close a session when there have been no open flows in either direction for a time. In another circumstance, an endpoint may be

ceasing operation and will close all of its sessions even if they have open flows.

To close an S\_OPEN session in a reliable and orderly fashion, an endpoint moves the session to the S\_NEARCLOSE state.

On a session transitioning from S\_OPEN to S\_NEARCLOSE and every 5 seconds thereafter while still in the S\_NEARCLOSE state, send a Session Close Request chunk (Section 2.3.17).

A session that has been in the S\_NEARCLOSE state for at least 90 seconds (allowing time to retransmit the Session Close Request multiple times) SHOULD move to the S\_CLOSED state.

On a session transitioning from S\_OPEN to the S\_NEARCLOSE, S\_FARCLOSE\_LINGER or S\_CLOSED state: immediately abort and terminate all open or closing flows. Flows only exist in S\_OPEN sessions.

To close an S\_OPEN session abruptly, send a Session Close Acknowledgement chunk (Section 2.3.18), then move to the S\_CLOSED state.

On receipt of a Session Close Request chunk for a session in the S\_OPEN, S\_NEARCLOSE, or S\_FARCLOSE\_LINGER states: send a Session Close Acknowledgement chunk; then, if the session is in the S\_OPEN state: move to the S\_FARCLOSE\_LINGER state.

A session that has been in the S\_FARCLOSE\_LINGER state for at least 19 seconds (allowing time to answer 3 retransmissions of a Session Close Request) SHOULD move to the S\_CLOSED state.

On receipt of a Session Close Acknowledgement chunk for a session in the S\_OPEN, S\_NEARCLOSE, or S\_FARCLOSE\_LINGER states: move to the S\_CLOSED state.

### 3.6. Flows

A flow is a unidirectional communication channel in a session for transporting a correlated series of user messages from a sender to a receiver. Each end of a session may have zero or more sending flows to the other end. Each sending flow at one end has a corresponding receiving flow at the other end.

#### 3.6.1. Overview

#### 3.6.1.1. Identity

Flows are multiplexed in a session by a flow identifier. Each end of a session chooses its sending flow identifiers independently of the other end. The choice of similar flow identifiers by both ends does not imply an association. A sender MAY choose any identifier for any flow; therefore, a flow receiver MUST NOT ascribe any semantic meaning, role, or name to a flow based only on its identifier. There are no "well known" or reserved flow identifiers.

Bidirectional flow association is indicated at flow startup with the Return Flow Association option (Section 2.3.11.1.2). An endpoint can indicate that a new sending flow is in return (or response) to a receiving flow from the other end. A sending flow MUST NOT indicate more than one return association. A receiving flow can be specified as the return association for any number of sending flows. The return flow association, if any, is fixed for the lifetime of the sending flow. Note: Closure of one flow in an association does not automatically close other flows in the association except as specified in Section 3.6.3.1.

Flows are named with arbitrary user metadata. This specification doesn't mandate any particular encoding, syntax, or semantics for the user metadata except for the encoded size (Section 2.3.11.1.1); the user metadata is entirely reserved for the application. The user metadata is fixed for the lifetime of the flow.

#### 3.6.1.2. Messages and Sequencing

Flows provide message-oriented framing. Large messages are fragmented for transport in the network. Receivers reassemble fragmented messages and only present complete messages to the user.

A sender queues messages on a sending flow one after another. A receiver can recover the original queuing order of the messages, even when they are reordered in transit by the network or as a result of loss and retransmission, by means of the messages' fragment sequence numbers. Flows are the basic units of message sequencing; each flow is sequenced independently of all other flows; inter-flow message arrival and delivery sequencing is not guaranteed.

Independent flow sequencing allows a sender to prioritize the transmission or retransmission of the messages of one flow over those of other flows in a session, allocating capacity from the transmission budget according to priority. RTMFP is designed for flows to be the basic unit of prioritization. In any flow, fragment sequence numbers are unique and monotonically increasing; that is, the fragment sequence numbers for any message MUST be greater than

the fragment sequence numbers of all messages previously queued in that flow. Receipt of fragments within a flow out of sequence number order creates discontinuous gaps at the receiver, causing it to send an acknowledgement for every packet and for the size of the encoded acknowledgements to grow. Therefore, for any flow, the sender SHOULD send lower sequence numbers first.

A sender can abandon a queued message at any time, even if some fragments of that message have been received by the other end. A receiver MUST be able to detect a gap in the flow when a message is abandoned; therefore, each message SHOULD take at least one sequence number from the sequence space even if no fragments for that message are ever sent. The sender will transmit the fragments of all messages not abandoned, and retransmit any lost fragments of all messages not abandoned, until all the fragments of all messages not abandoned are acknowledged by the receiver. A sender indicates a Forward Sequence Number (FSN) to instruct the receiver that sequence numbers less than or equal to the FSN will not be transmitted or retransmitted. This allows the receiver to move forward over gaps and continue sequenced delivery of completely received messages to the user. Any incomplete messages missing fragments with sequence numbers less than or equal to the FSN were abandoned by the sender and will never be completed. A gap indication MUST be communicated to the receiving user.

#### 3.6.1.3. Lifetime

A sender begins a flow by sending user message fragments to the other end, and including the user metadata and, if any, the return flow association. The sender continues to include the user metadata and return flow association until the flow is acknowledged by the far end, at which point the sender knows that the receiver has received the user metadata and, if any, the return flow association. After that point, the flow identifier alone is sufficient.

Flow receivers SHOULD acknowledge all sequence numbers received for any flow, whether the flow is accepted or rejected. Flow receivers MUST NOT acknowledge sequence numbers higher than the FSN that were not received. Acknowledgements drive the congestion control and avoidance algorithms and therefore must be accurate.

An endpoint can reject a receiving flow at any time in the flow's lifetime. To reject the flow, the receiving endpoint sends a Flow Exception chunk (Section 2.3.16) immediately preceding every acknowledgement chunk for the rejected receiving flow.

An endpoint may eventually conclude and close a sending flow. The last sequence number of the flow is marked with the Final flag. The



sending flow is complete when all sequence numbers of the flow, including the final sequence number, have been cumulatively acknowledged by the receiver. The receiving flow is complete when every sequence number from the FSN to the final sequence number has been received. The sending flow and corresponding receiving flow at the respective ends hold the flow identifier of a completed flow in reserve for a time to allow delayed or duplicated fragments and acknowledgements to drain from the network without erroneously initiating a new receiving flow or erroneously acknowledging a new sending flow.

If a flow sender receives a Flow Exception indication from the other end, the flow sender SHOULD close the flow and abandon all of the undelivered queued messages. The flow sender SHOULD indicate an exception to the user.

### 3.6.2. Sender

Each sending flow comprises the flow-specific information context necessary to transfer that flow's messages to the other end. Each sending flow context includes at least:

- o F\_FLOW\_ID: this flow's identifier;
- o STARTUP\_OPTIONS: the set of options to send to the receiver until this flow is acknowledged, including the User's Per-Flow Metadata and, if set, the Return Flow Association;
- o SEND\_QUEUE: the unacknowledged message fragments queued in this flow, initially empty; each message fragment entry comprising:
  - \* SEQUENCE\_NUMBER: the sequence number of this fragment;
  - \* DATA: this fragment's user data;
  - \* FRA: the fragment control value for this message fragment, having one of the values enumerated for that purpose in User Data (Section 2.3.11);
  - \* ABANDONED: a boolean flag indicating whether this fragment has been abandoned;
  - \* SENT\_ABANDONED: a boolean flag indicating whether this fragment was abandoned when sent;
  - \* EVER\_SENT: a boolean flag indicating whether this fragment has been sent at least once, initially false;

- \* NAK\_COUNT: a count of the number of negative acknowledgements detected for this fragment, initially 0;
  - \* IN\_FLIGHT: a boolean flag indicating whether this fragment is currently outstanding, or in flight, in the network, initially false;
  - \* TRANSMIT\_SIZE: the size, in bytes, of the encoded User Data chunk (including the chunk header) for this fragment when it was transmitted into the network.
- o F\_OUTSTANDING\_BYTES: the sum of the TRANSMIT\_SIZE of each entry in SEND\_QUEUE where entry.IN\_FLIGHT is true;
  - o RX\_BUFFER\_SIZE: the most recent available buffer advertisement from the other end (Section 2.3.13, Section 2.3.14), initially 65536 bytes;
  - o NEXT\_SN: the next sequence number to assign to a message fragment, initially 1;
  - o F\_FINAL\_SN: the sequence number assigned to the final message fragment of the flow, initially having no value;
  - o EXCEPTION: a boolean flag indicating whether an exception has been reported by the receiver, initially false;
  - o The state, at any time being one of the following values: the open state F\_OPEN; the closing states F\_CLOSING and F\_COMPLETE\_LINGER; and the closed state F\_CLOSED.

Note: this diagram is only a summary of state transitions and their causing events, and is not a complete operational specification.

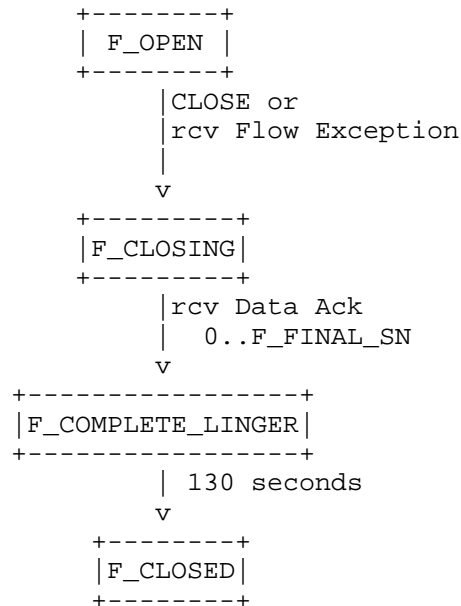


Figure 19: Sending flow state diagram

#### 3.6.2.1. Startup

The application opens a new sending flow to the other end in an S\_OPEN session. The implementation chooses a new flow ID that is not assigned to any other sending flow in that session in the F\_OPEN, F\_CLOSING, or F\_COMPLETE\_LINGER states. The flow starts in the F\_OPEN state. The STARTUP\_OPTIONS for the new flow is set with the User's Per-Flow Metadata (Section 2.3.11.1.1). If this flow is in return (or response) to a receiving flow from the other end, that flow's ID is encoded in a Return Flow Association (Section 2.3.11.1.2) option and added to STARTUP\_OPTIONS. A new sending flow SHOULD NOT be opened in response to a receiving flow from the other end that is not in the RF\_OPEN state when the sending flow is opened.

At this point the flow exists in the sender, but not the receiver. The flow begins when user data fragments are transmitted to the receiver. A sender can begin a flow in the absence of immediate user data by sending a Forward Sequence Number Update (Section 3.6.2.7.1), by queuing and transmitting a user data fragment that is already abandoned.

### 3.6.2.2. Queuing Data

The application queues messages in an F\_OPEN sending flow for transmission to the far end. The implementation divides each message into one or more fragments for transmission in User Data chunks (Section 2.3.11). Each fragment MUST be small enough so that, if assembled into a Packet (Section 2.2.4) with a maximum-size common header, User Data chunk header, and, if not empty, this flow's STARTUP\_OPTIONS, the Packet will not exceed the Path MTU (Section 3.5.4.3).

For each fragment, create a fragment entry and set fragmentEntry.SEQUENCE\_NUMBER to flow.NEXT\_SN, and increment flow.NEXT\_SN by one. Set fragmentEntry.FRA according to the encoding in User Data chunks:

- 0 : This fragment is a complete message.
- 1 : This fragment is the first of a multi-fragment message.
- 2 : This fragment is the last of a multi-fragment message.
- 3 : This fragment is in the middle of a multi-fragment message.

Append fragmentEntry to flow.SEND\_QUEUE.

### 3.6.2.3. Sending Data

A sending flow is ready to transmit if the SEND\_QUEUE contains at least one entry that is eligible to send and if either RX\_BUFFER\_SIZE is greater than F\_OUTSTANDING\_BYTES or EXCEPTION is set to true.

A SEND\_QUEUE entry is eligible to send if it is not IN\_FLIGHT AND at least one of the following conditions holds:

- o The entry is not ABANDONED; or
- o The entry is the first one in the SEND\_QUEUE; or
- o The entry's SEQUENCE\_NUMBER is equal to flow.F\_FINAL\_SN.

If the session's transmission budget allows, a flow that is ready to transmit is selected for transmission according to the implementation's prioritization scheme. The manner of flow prioritization is not mandated by this specification.

Trim abandoned messages from the front of the queue and find the Forward Sequence Number FSN:

1. While the SEND\_QUEUE contains at least two entries, AND the first entry is not IN\_FLIGHT, AND the first entry is ABANDONED: remove and discard the first entry from SEND\_QUEUE;
2. If the first entry in the SEND\_QUEUE is not abandoned: set FSN to entry.SEQUENCE\_NUMBER - 1; otherwise
3. If the first entry in the SEND\_QUEUE is IN\_FLIGHT AND entry.SENT\_ABANDONED is false: set FSN to entry.SEQUENCE\_NUMBER - 1; otherwise
4. The first entry in the SEND\_QUEUE is abandoned and is either not IN\_FLIGHT or was already abandoned when sent: set FSN to entry.SEQUENCE\_NUMBER.

The FSN MUST NOT be greater than any sequence number currently outstanding. The FSN MUST NOT be equal to any sequence number currently outstanding that was not abandoned when sent.

Assemble user data chunks for this flow into a packet to send to the receiver. While enough space remains in the packet and the flow is ready to transmit:

1. Starting at the head of the SEND\_QUEUE, find the first eligible fragment entry;
2. Encode entry into a User Data chunk (Section 2.3.11) or, if possible (Section 3.6.2.3.2), a Next User Data chunk (Section 2.3.12);
3. If present, set chunk.flowID to flow.F\_FLOW\_ID;
4. If present, set chunk.sequenceNumber to entry.SEQUENCE\_NUMBER;
5. If present, set chunk.fsnOffset to entry.SEQUENCE\_NUMBER - FSN;
6. Set chunk.fragmentControl to entry.FRA;
7. Set chunk.abandon to entry.ABANDONED;
8. If entry.SEQUENCE\_NUMBER equals flow.F\_FINAL\_SN: set chunk.final to true; else set chunk.final to false;
9. If any options are being sent with this chunk: set chunk.optionsPresent to true, assemble the options into the chunk, and assemble a Marker to terminate the option list;

10. If entry.ABANDONED is true, set chunk.userData to empty;  
otherwise set chunk.userData to entry.DATA;
11. If adding the assembled chunk to the packet would cause the  
packet to exceed the path MTU: do not assemble this chunk into  
the packet, enough space no longer remains in the packet, stop.  
Otherwise, continue:
12. Set entry.IN\_FLIGHT to true;
13. Set entry.EVER\_SENT to true;
14. Set entry.NAK\_COUNT to 0;
15. Set entry.SENT\_ABANDONED to entry.ABANDONED;
16. Set entry.TRANSMIT\_SIZE to the size of the assembled chunk,  
including the chunk header;
17. Assemble this chunk into the packet; and
18. If this flow or entry is considered Time Critical (real-time),  
set the timeCritical flag in the packet header (Section 2.2.4).

Complete any other appropriate packet processing, and transmit the  
packet to the far end.

#### 3.6.2.3.1. Startup Options

If STARTUP\_OPTIONS is not empty, then when assembling the FIRST User  
Data chunk for this flow into a packet, add the encoded  
STARTUP\_OPTIONS to that chunk's option list.

#### 3.6.2.3.2. Send Next Data

The Next User Data chunk (Section 2.3.12) is a compact encoding for a  
user message fragment when multiple contiguous fragments are  
assembled into one packet. Using this chunk where possible can  
conserve space in a packet, potentially reducing transmission  
overhead or allowing additional information to be sent in a packet.

If, after assembling a user message fragment of a flow into a packet  
(Section 3.6.2.3), the next eligible fragment to be selected for  
assembly into that packet belongs to the same flow AND its sequence  
number is one greater than that of the fragment just assembled, it is  
RECOMMENDED that an implementation encode a Next User Data chunk  
instead of a User Data chunk.

The FIRST fragment of a flow assembled into a packet MUST be encoded as a User Data chunk.

#### 3.6.2.4. Processing Acknowledgements

A Data Acknowledgement Bitmap chunk (Section 2.3.13) or a Data Acknowledgement Ranges chunk (Section 2.3.14) encodes the acknowledgement of receipt of one or more sequence numbers of a flow, as well as the receiver's current receive window advertisement.

On receipt of an acknowledgement chunk for a sending flow:

1. Set PRE\_ACK\_OUTSTANDING\_BYTES to flow.F\_OUTSTANDING\_BYTES;
2. Set flow.STARTUP\_OPTIONS to empty;
3. Set flow.RX\_BUFFER\_SIZE to chunk.bufferBytesAvailable;
4. For each sequence number encoded in the acknowledgement: if there is an entry in flow.SEND\_QUEUE with that sequence number and its IN\_FLIGHT is true, then: remove entry from flow.SEND\_QUEUE; and
5. Notify the congestion control and avoidance algorithms that PRE\_ACK\_OUTSTANDING\_BYTES - flow.F\_OUTSTANDING\_BYTES were acknowledged. Note that Negative Acknowledgements (Section 3.6.2.5) affect "TCP friendly" congestion control.

#### 3.6.2.5. Negative Acknowledgement and Loss

A negative acknowledgement is inferred for an outstanding fragment if an acknowledgement is received for any other fragments sent after it in the same session.

An implementation SHOULD consider a fragment to be lost once that fragment receives three negative acknowledgements. A lost fragment is no longer outstanding in the network.

The following describes an OPTIONAL method for detecting negative acknowledgements.

Let the session track the order in which fragments are transmitted across all its sending flows by way of a monotonically increasing Transmission Sequence Number (TSN) recorded with each fragment queue entry each time that fragment is transmitted.

Let the session information context contain additional variables:

- o `NEXT_TSN`: the next TSN to record with a fragment's queue entry when it is transmitted, initially 1;
- o `MAX_TSN_ACK`: the highest acknowledged TSN, initially 0.

Let each fragment queue entry contain an additional variable `TSN`, initially 0, to track its transmission order.

On transmission of a message fragment into the network, set its `entry.TSN` to `session.NEXT_TSN`, and increment `session.NEXT_TSN`.

On acknowledgement of an outstanding fragment, if its `entry.TSN` is greater than `session.MAX_TSN_ACK`, set `session.MAX_TSN_ACK` to `entry.TSN`.

After processing all acknowledgements in a packet containing at least one acknowledgement, then for each sending flow in that session, for each entry in that flow's `SEND_QUEUE`, if `entry.IN_FLIGHT` is true and `entry.TSN` is less than `session.MAX_TSN_ACK`: increment `entry.NAK_COUNT` and notify the congestion control and avoidance algorithms that a negative acknowledgement was detected in this packet.

For each sending flow in that session, for each entry in that flow's `SEND_QUEUE`, if `entry.IN_FLIGHT` is true and `entry.NAK_COUNT` is at least 3, that fragment was lost in the network and is no longer considered to be in flight. Set `entry.IN_FLIGHT` to false. Notify the congestion control and avoidance algorithms of the loss.

#### 3.6.2.6. Timeout

A fragment is considered lost and no longer in flight in the network if it has remained outstanding for at least `ERTO`.

The following describes an OPTIONAL method to manage transmission timeouts. This method REQUIRES that either Burst Avoidance (Section 3.5.2.3) is implemented, or that the implementation's congestion control and avoidance algorithms will eventually stop sending new fragments into the network if acknowledgements are persistently not received.

Let the session information context contain an alarm `TIMEOUT_ALARM`, initially unset.

On sending a packet containing at least one User Data chunk, set or reset `TIMEOUT_ALARM` to fire in `ERTO`.

On receiving a packet containing at least one acknowledgement, reset `TIMEOUT_ALARM` (if already set) to fire in `ERTO`.



When TIMEOUT\_ALARM fires:

1. Set WAS\_LOSS = false;
2. For each sending flow in the session, and for each entry in that flow's SEND\_QUEUE:
  1. If entry.IN\_FLIGHT is true: set WAS\_LOSS = true; and
  2. Set entry.IN\_FLIGHT to false.
3. If WAS\_LOSS is true: perform ERTT backoff (Section 3.5.2.2); and
4. Notify the congestion control and avoidance algorithms of the timeout and, if WAS\_LOSS is true, that there was loss.

#### 3.6.2.7. Abandoning Data

The application can abandon queued messages at any time and for any reason. Example reasons include (but are not limited to): one or more fragments of a message have remained in the SEND\_QUEUE for longer than a specified message lifetime; a fragment has been retransmitted more than a specified retransmission limit; a prior message on which this message depends (such as a key frame in a prediction chain) was abandoned and not delivered.

To abandon a message fragment, set its SEND\_QUEUE entry's ABANDON flag to true. When abandoning a message fragment, abandon all fragments of the message to which it belongs.

An abandoned fragment MUST NOT be un-abandoned.

##### 3.6.2.7.1. Forward Sequence Number Update

Abandoned data may leave gaps in the sequence number space of a flow. Gaps may cause the receiver to hold completely received messages for ordered delivery to allow for retransmission of the missing fragments. User Data chunks (Section 2.3.11) encode a Forward Sequence Number (FSN) to instruct the receiver that fragments with sequence numbers less than or equal to the FSN will not be transmitted or retransmitted.

When the receiver has gaps in the received sequence number space and no non-abandoned message fragments remain in the SEND\_QUEUE, the sender SHOULD transmit a Forward Sequence Number Update (FSN Update) comprising a User Data chunk marked abandoned, whose sequence number is the FSN and whose fsnOffset is 0. An FSN Update allows the receiver to skip gaps that will not be repaired and deliver received

messages to the user. An FSN Update may be thought of as a transmission or retransmission of abandoned sequence numbers without actually sending the data.

The method described in Sending Data (Section 3.6.2.3) generates FSN Updates when appropriate.

#### 3.6.2.8. Examples

```
Sender
|
1 |<--- Ack ID=2, seq:0-16
2 |---> Data ID=2, seq#=25, fsnOff=9 (fsn=16)
3 |---> Data ID=2, seq#=26, fsnOff=10 (fsn=16)
4 |<--- Ack ID=2, seq:0-18
5 |---> Data ID=2, seq#=27, fsnOff=9 (fsn=18)
6 |---> Data ID=2, seq#=28, fsnOff=10 (fsn=18)
|
:
```

There are 9 sequence numbers in flight with delayed acknowledgements.

Figure 20: Normal flow with no loss

```

Sender
:
1  <--- Ack   ID=3, seq:0-30
2  ---> Data ID=3, seq#=45, fsnOff=15 (fsn=30)
3  <--- Ack   ID=3, seq:0-30, 32 (nack 31:1)
4  ---> Data ID=3, seq#=46, fsnOff=16 (fsn=30)
5  <--- Ack   ID=3, seq:0-30, 32, 34 (nack 31:2, 33:1)
6  <--- Ack   ID=3, seq:0-30, 32, 34-35 (nack 31:3=lost, 33:2)
7  ---> Data ID=3, seq#=47, fsnOff=15 (fsn=32, abandon 31)
8  <--- Ack   ID=3, seq:0-30, 32, 34-36 (nack 33:3=lost)
9  ---> Data ID=3, seq#=33, fsnOff=1 (fsn=32, retransmit 33)
10 <--- Ack   ID=3, seq:0-30, 32, 34-37
11 ---> Data ID=3, seq#=48, fsnOff=16 (fsn=32)
:
   (continues through seq#=59)
:
12 ---> Data ID=3, seq#=60, fsnOff=28 (fsn=32)
13 <--- Ack   ID=3, seq:0-30, 34-46
14 ---> Data ID=3, seq#=61, fsnOff=29 (fsn=32)
15 <--- Ack   ID=3, seq:0-32, 34-47
16 ---> Data ID=3, seq#=62, fsnOff=30 (fsn=32)
17 <--- Ack   ID=3, seq:0-47
18 ---> Data ID=3, seq#=63, fsnOff=16 (fsn=47)
19 <--- Ack   ID=3, seq:0-49
20 ---> Data ID=3, seq#=64, fsnOff=15 (fsn=49)
:
21 <--- Ack   ID=3, seq:0-59
22 <--- Ack   ID=3, seq:0-59, 61 (nack 60:1)
23 <--- Ack   ID=3, seq:0-59, 61-62 (nack 60:2)
24 <--- Ack   ID=3, seq:0-59, 61-63 (nack 60:3=lost)
25 ---> Data ID=3, ABN=1, seq#=60, fsnOff=0 (fsn=60, abandon 60)
26 <--- Ack   ID=3, seq:0-59, 61-64
:
27 <--- Ack   ID=3, seq:0-64

```

Flow with sequence numbers 31, 33, and 60 lost in transit, and a pause at 64. 33 is retransmitted, 31 and 60 are abandoned. Note line 25 is a Forward Sequence Number Update (Section 3.6.2.7.1).

Figure 21: Flow with loss

### 3.6.2.9. Flow Control

The flow receiver advertises the amount of new data it's willing to accept from the flow sender with the `bufferBytesAvailable` derived field of an acknowledgement (Section 2.3.13, Section 2.3.14).

The flow sender MUST NOT send new data into the network if

flow.F\_OUTSTANDING\_BYTES is greater than or equal to the most recently received buffer advertisement, unless flow.EXCEPTION is true (Section 3.6.2.3).

#### 3.6.2.9.1. Buffer Probe

The flow sender is suspended if the most recently received buffer advertisement is zero and the flow hasn't been rejected by the receiver; that is, while RX\_BUFFER\_SIZE is zero AND EXCEPTION is false. To guard against potentially lost acknowledgements that might reopen the receive window, a suspended flow sender SHOULD send a packet comprising a Buffer Probe chunk (Section 2.3.15) for this flow from time to time.

If the receive window advertisement transitions from non-zero to zero, the flow sender MAY send a Buffer Probe immediately and SHOULD send a probe within one second.

The initial period between Buffer Probes SHOULD be at least one second or ERT0, whichever is greater. The period between probes SHOULD increase over time, but the period between probes SHOULD NOT be more than one minute or ERT0, whichever is greater.

The flow sender SHOULD stop sending Buffer Probes if it is no longer suspended.

#### 3.6.2.10. Exception

The flow receiver can reject the flow at any time and for any reason. The flow receiver sends a Flow Exception Report (Section 2.3.16) when it has rejected a flow.

On receiving a Flow Exception Report for a sending flow:

1. If the flow is F\_OPEN, close the flow (Section 3.6.2.11) and notify the user that the far end reported an exception with the encoded exception code;
2. Set the EXCEPTION flag to true; and
3. For each entry in SEND\_QUEUE: set entry.ABANDONED = true.

#### 3.6.2.11. Close

A sending flow is closed by the user or as a result of an exception. To close an F\_OPEN flow:

1. Move to the F\_CLOSING state;
2. If the SEND\_QUEUE is not empty, AND the tail entry of the SEND\_QUEUE has a sequence number of NEXT\_SN - 1, AND the tail entry.EVER\_SENT is false: set F\_FINAL\_SN to entry.SEQUENCE\_NUMBER; else
3. The SEND\_QUEUE is empty, OR the tail entry does not have a sequence number of NEXT\_SN - 1, OR the tail entry.EVER\_SENT is true: enqueue a new SEND\_QUEUE entry with entry.SEQUENCE\_NUMBER = flow.NEXT\_SN, entry.FRA = 0, and entry.ABANDONED = true, and set flow.F\_FINAL\_SN to entry.SEQUENCE\_NUMBER.

An F\_CLOSING sending flow is complete when its SEND\_QUEUE transitions to empty, indicating that all sequence numbers including the FINAL\_SN have been acknowledged by the other end.

When an F\_CLOSING sending flow becomes complete, move to the F\_COMPLETE\_LINGER state.

A sending flow MUST remain in the F\_COMPLETE\_LINGER state for at least 130 seconds. After at least 130 seconds, move to the F\_CLOSED state. The sending flow is now closed, its resources can be reclaimed, and its F\_FLOW\_ID MAY be used for a new sending flow.

### 3.6.3. Receiver

Each receiving flow comprises the flow-specific information context necessary to receive that flow's messages from the sending end and deliver completed messages to the user. Each receiving flow context includes at least:

- o RF\_FLOW\_ID: this flow's identifier;
- o SEQUENCE\_SET: the set of all fragment sequence numbers seen in this receiving flow, whether received or abandoned, initially empty;
- o RF\_FINAL\_SN: the final fragment sequence number of the flow, initially having no value;
- o RECV\_BUFFER: the message fragments waiting to be delivered to the user, sorted by sequence number in ascending order, initially empty; each message fragment entry comprising:
  - \* SEQUENCE\_NUMBER: the sequence number of this fragment;

- \* DATA: this fragment's user data; and
- \* FRA: the fragment control value for this message fragment, having one of the values enumerated for that purpose in User Data (Section 2.3.11).
- o BUFFERED\_SIZE: the sum of the lengths of each fragment in RECV\_BUFFER plus any additional storage overhead for the fragments incurred by the implementation, in bytes;
- o BUFFER\_CAPACITY: the desired maximum size for the receive buffer, in bytes;
- o PREV\_RWND: the most recent receive window advertisement sent in an acknowledgement, in 1024-byte blocks, initially having no value;
- o SHOULD\_ACK: whether or not an acknowledgement should be sent for this flow, initially false;
- o EXCEPTION\_CODE: the exception code to report to the sender when the flow has been rejected, initially 0;
- o The state, at any time being one of the following values: the open state RF\_OPEN; the closing states RF\_REJECTED and RF\_COMPLETE\_LINGER; and the closed state RF\_CLOSED.

Note: this diagram is only a summary of state transitions and their causing events, and is not a complete operational specification.

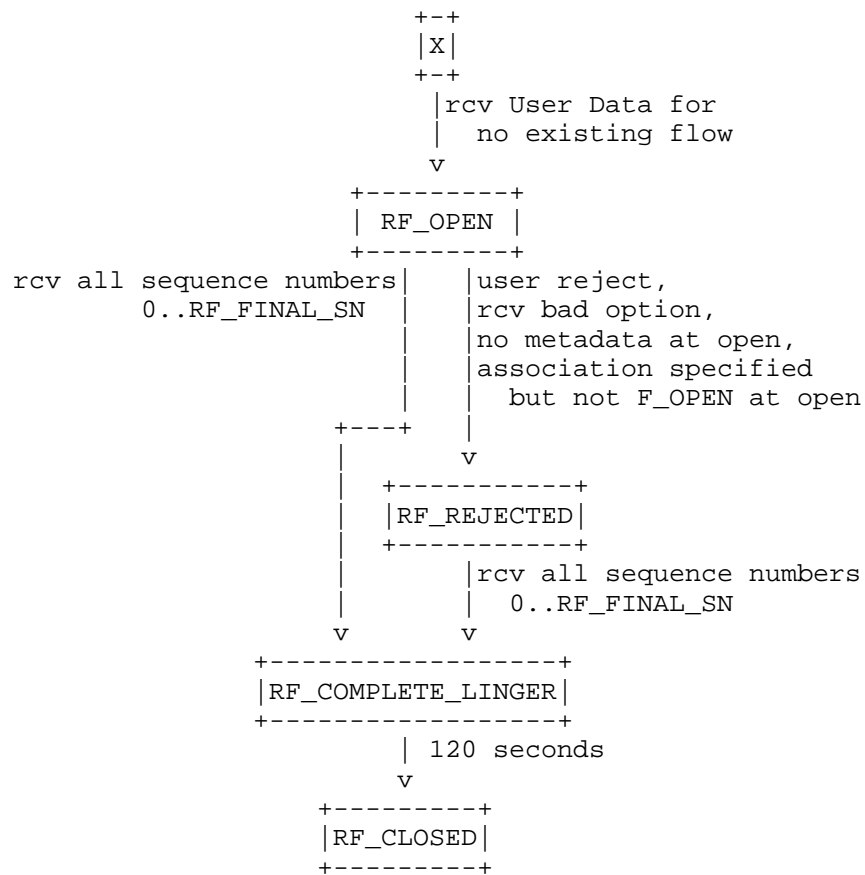


Figure 22: Receiving flow state diagram

#### 3.6.3.1. Startup

A new receiving flow starts on receipt of a User Data chunk (Section 2.3.11) encoding a flow ID not belonging to any other receiving flow in the same session in the RF\_OPEN, RF\_REJECTED, or RF\_COMPLETE\_LINGER states.

On receipt of such a User Data chunk:

1. Set temporary variables METADATA, ASSOCIATED\_FLOWID, and ASSOCIATION to each have no value;

2. Create a new receiving flow context in this session, setting its RF\_FLOW\_ID to the flow ID encoded in the opening User Data chunk, and set to the RF\_OPEN state;
3. If the opening User Data chunk encodes a User's Per-Flow Metadata option (Section 2.3.11.1.1), set METADATA to option.userMetadata;
4. If the opening User Data chunk encodes a Return Flow Association option (Section 2.3.11.1.2), set ASSOCIATED\_FLOWID to option.flowID;
5. If METADATA has no value, the receiver MUST reject the flow (Section 3.6.3.7), moving it to the RF\_REJECTED state;
6. If ASSOCIATED\_FLOWID has a value, then if there is no sending flow in the same session with a flow ID of ASSOCIATED\_FLOWID, the receiver MUST reject the flow, moving it to the RF\_REJECTED state; otherwise set ASSOCIATION to the indicated sending flow;
7. If ASSOCIATION indicates a sending flow AND that sending flow's state is not F\_OPEN, the receiver MUST reject this receiving flow, moving it to the RF\_REJECTED state;
8. If the opening User Data chunk encodes any unrecognized option with a type code less than 8192 (Section 2.3.11.1), the receiver MUST reject the flow, moving it to the RF\_REJECTED state;
9. If this new receiving flow is still RF\_OPEN, then: notify the user that a new receiving flow has opened, including the METADATA and, if present, the ASSOCIATION, and set flow.BUFFER\_CAPACITY according to the user;
10. Perform the normal data processing (Section 3.6.3.2) for the opening User Data chunk; and
11. Set this session's ACK\_NOW to true.

#### 3.6.3.2. Receiving Data

A User Data chunk (Section 2.3.11) or a Next User Data chunk (Section 2.3.12) encodes one fragment of a user data message of a flow, as well as the flow's Forward Sequence Number and potentially optional parameters (Section 2.3.11.1).

On receipt of a User Data or Next User Data chunk:



1. If chunk.flowID doesn't indicate an existing receiving flow in the same session in the RF\_OPEN, RF\_REJECTED, or RF\_COMPLETE\_LINGER state, perform the steps at Startup (Section 3.6.3.1) to start a new receiving flow;
2. Retrieve the receiving flow context for the flow indicated by chunk.flowID;
3. Set flow.SHOULD\_ACK to true;
4. If the flow is RF\_OPEN AND the chunk encodes any unrecognized option with a type code less than 8192 (Section 2.3.11.1), the flow MUST be rejected: notify the user of an exception, and reject the flow (Section 3.6.3.7), moving it to the RF\_REJECTED state;
5. If the flow is not in the RF\_OPEN state: set session.ACK\_NOW to true;
6. If flow.PREV\_RWND has a value and that value is less than 2 blocks, set session.ACK\_NOW to true;
7. If chunk.abandon is true: set session.ACK\_NOW to true;
8. If flow.SEQUENCE\_SET has any gaps (that is, if it doesn't contain every sequence number from 0 through and including the highest sequence number in the set), set session.ACK\_NOW to true;
9. If flow.SEQUENCE\_SET contains chunk.sequenceNumber, then this chunk is a duplicate: set session.ACK\_NOW to true;
10. If flow.SEQUENCE\_SET doesn't contain chunk.sequenceNumber, AND chunk.final is true, AND flow.RF\_FINAL\_SN has no value, then: set flow.RF\_FINAL\_SN to chunk.sequenceNumber, and set session.ACK\_NOW to true;
11. If the flow is in the RF\_OPEN state, AND flow.SEQUENCE\_SET doesn't contain chunk.sequenceNumber, AND chunk.abandon is false, then: create a new RECV\_BUFFER entry for this chunk's data and set entry.SEQUENCE\_NUMBER to chunk.sequenceNumber, entry.DATA to chunk.userData, and entry.FRA to chunk.fragmentControl, and insert this new entry into flow.RECV\_BUFFER;
12. Add to flow.SEQUENCE\_SET the range of sequence numbers from 0 through and including the chunk.forwardSequenceNumber derived field;

13. Add `chunk.sequenceNumber` to `flow.SEQUENCE_SET`;
14. If `flow.SEQUENCE_SET` now has any gaps, set `session.ACK_NOW` to `true`;
15. If `session.ACK_NOW` is false and `session.DELACK_ALARM` is not set: set `session.DELACK_ALARM` to fire in 200 milliseconds; and
16. Attempt delivery of completed messages in this flow's `RECV_BUFFER` to the user (Section 3.6.3.3).

After processing all chunks in a packet containing at least one User Data chunk, increment `session.RX_DATA_PACKETS` by one. If `session.RX_DATA_PACKETS` is at least two, set `session.ACK_NOW` to `true`.

A receiving flow that is not in the `RF_CLOSED` state is ready to send an acknowledgement if its `SHOULD_ACK` flag is set. Acknowledgements for receiving flows that are ready are sent either opportunistically by piggybacking on a packet that's already sending user data or an acknowledgement (Section 3.6.3.4.6), or when the session's `ACK_NOW` flag is set (Section 3.6.3.4.5).

#### 3.6.3.3. Buffering and Delivering Data

A receiving flow's information context contains a `RECV_BUFFER` for reordering, reassembling, and holding the user data messages of the flow. Only complete messages are delivered to the user; an implementation **MUST NOT** deliver partially received messages except by special arrangement with the user.

Let the Cumulative Acknowledgement Sequence Number CSN be the highest number in the contiguous range of numbers in `SEQUENCE_SET` starting with 0. For example, if `SEQUENCE_SET` contains {0, 1, 2, 3, 5, 6}, the contiguous range starting with 0 is 0..3, so the CSN is 3.

A message is complete if all of its fragments are present in the `RECV_BUFFER`. The fragments of one message have contiguous sequence numbers. A message can either be a single fragment, whose fragment control value is 0-whole, or can be two or more fragments where the first's fragment control value is 1-begin, followed by zero or more fragments with control value 3-middle, and terminated by a last fragment with control value 2-end.

An incomplete message segment is a contiguous sequence of one or more fragments that do not form a complete message; that is, a 1-begin followed by zero or more 3-middle fragments but with no 2-end, or zero or more 3-middle fragments followed by a 2-end but with no 1-begin, or one or more 3-middle fragments with neither a 1-begin nor

a 2-end.

Incomplete message segments can either be in progress or abandoned. An incomplete segment is abandoned in the following cases:

- o The sequence number of the segment's first fragment is less than or equal to the CSN AND that fragment's control value is not 1-begin; or
- o The sequence number of the segment's last fragment is less than the CSN.

Abandoned message segments will never be completed, so they SHOULD be removed from the RECV\_BUFFER to make room in the advertised receive window and the receiver's memory for messages that can be completed.

The user can suspend delivery of a flow's messages. A suspended receiving flow holds completed messages in its RECV\_BUFFER until the user resumes delivery. A suspended flow can cause the receive window advertisement to go to zero even when the BUFFER\_CAPACITY is non-zero; this is described in detail in Flow Control (Section 3.6.3.5).

When the receiving flow is not suspended, the original queuing order of the messages is recovered by delivering, in ascending sequence number order, complete messages in the RECV\_BUFFER whose sequence numbers are less than or equal to the CSN.

The following describes a method for discarding abandoned message segments and delivering complete messages in original queueing order when the receiving flow is not suspended.

While the first fragment entry in the RECV\_BUFFER has a sequence number less than or equal to CSN and delivery is still possible:

1. If entry.FRA is 0-whole: deliver entry.DATA to the user, and remove this entry from RECV\_BUFFER; otherwise,
2. If entry.FRA is 2-end or 3-middle: this entry belongs to an abandoned segment, so remove and discard this entry from RECV\_BUFFER; otherwise,
3. Entry.FRA is 1-begin. Let LAST\_ENTRY be the last RECV\_BUFFER entry that is part of this message segment (LAST\_ENTRY can be entry if the segment has only one fragment so far). Then:
  1. If LAST\_ENTRY.FRA is 2-end: this segment is a complete message, so concatenate the DATA fields of each fragment entry of this segment in ascending sequence number order and

deliver the complete message to the user, then remove the entries for this complete message from RECV\_BUFFER; otherwise,

2. If LAST\_ENTRY.SEQUENCE\_NUMBER is less than CSN: this segment is incomplete and abandoned, so remove and discard the entries for this segment from RECV\_BUFFER; otherwise,
3. LAST\_ENTRY.SEQUENCE\_NUMBER is equal to CSN and LAST\_ENTRY.FRA is not 2-end: this segment is incomplete but still in progress. Ordered delivery is no longer possible until at least one more fragment is received. Stop.

If flow.RF\_FINAL\_SN has a value and is equal to CSN, AND RECV\_BUFFER is empty: all complete messages have been delivered to the user, so notify the user that the flow is complete.

#### 3.6.3.4. Acknowledging Data

A flow receiver SHOULD acknowledge all user data fragment sequence numbers seen in that flow. Acknowledgements drive the sender's congestion control and avoidance algorithms, clear data from the sender's buffers, and in some sender implementations clock new data into the network, and therefore must be accurate and timely.

##### 3.6.3.4.1. Timing

For similar reasons as discussed in RFC 1122 Section 4.2.3.2 [RFC1122], it is advantageous to delay sending acknowledgements for a short time so that multiple data fragments can be acknowledged in a single transmission. However, it is also advantageous for a sender to receive timely notification about the receiver's disposition of the flow, particularly in unusual or exceptional circumstances, so that the circumstances can be addressed if possible.

Therefore, a flow receiver SHOULD send an acknowledgement for a flow as soon as is practical in any of the following circumstances:

- o On receipt of a User Data chunk that starts a new flow;
- o On receipt of a User Data or Next User Data chunk if the flow is not in the RF\_OPEN state;
- o On receipt of a User Data chunk where, before processing the chunk, the SEQUENCE\_SET of the indicated flow does not contain every sequence number between 0 and the highest sequence number in the set (that is, if there was a sequence number gap before processing the chunk);

- o On receipt of a User Data chunk where, after processing the chunk, the flow's SEQUENCE\_SET does not contain every sequence number between 0 and the highest sequence number in the set (that is, if this chunk causes a sequence number gap);
- o On receipt of a Buffer Probe for the flow;
- o On receipt of a User Data chunk if the last acknowledgement sent for the flow indicated fewer than two bufferBlocksAvailable;
- o On receipt of a User Data or Next User Data chunk for the flow if, after processing the chunk, the flow's BUFFER\_CAPACITY is not at least 1024 bytes greater than BUFFERED\_SIZE;
- o On receipt of a User Data or Next User Data chunk for any sequence number that was already seen (that is, on receipt of a duplicate);
- o On the first receipt of the final sequence number of the flow;
- o On receipt of two packets in the session containing user data for any flows since an acknowledgement was last sent; the new acknowledgements being for the flows having any User Data chunks in the received packets (that is, for every second packet containing user data);
- o After receipt of a User Data chunk for the flow, if an acknowledgement for any other flow is being sent (that is, consolidate acknowledgements);
- o After receipt of a User Data chunk for the flow, if any user data for a sending flow is being sent in a packet and if there is space available in the same packet (that is, attempt to piggyback an acknowledgement with user data if possible);
- o No longer than 200 milliseconds after receipt of a User Data chunk for the flow.

#### 3.6.3.4.2. Size and Truncation

Including an encoded acknowledgement in a packet might cause the packet to exceed the path MTU. In that case:

- o If the packet is being sent primarily to send an acknowledgement AND this is the first acknowledgement in the packet, truncate the acknowledgement so that the packet does not exceed the path MTU; otherwise

- o The acknowledgement is being piggybacked in a packet with user data or with an acknowledgement for another flow: do not include this acknowledgement in the packet, and send it later.

#### 3.6.3.4.3. Constructing

The Data Acknowledgement Bitmap chunk (Section 2.3.13) and Data Acknowledgement Ranges chunk (Section 2.3.14) encode a receiving flow's `SEQUENCE_SET` and its receive window advertisement. The two chunks are semantically equivalent; implementations **SHOULD** send whichever provides the most compact encoding of the `SEQUENCE_SET`.

When assembling an acknowledgement for a receiving flow:

1. If the flow's state is `RF_REJECTED`, first assemble a Flow Exception Report chunk (Section 2.3.16) for `flow.flowID`;
2. Choose the acknowledgement chunk type that most compactly encodes `flow.SEQUENCE_SET`;
3. Use the method described in Flow Control (Section 3.6.3.5) to determine the value for the acknowledgement chunk's `bufferBlocksAvailable` field;

#### 3.6.3.4.4. Delayed Acknowledgement

As discussed in Acknowledging Data (Section 3.6.3.4.1), a flow receiver can delay sending an acknowledgement for up to 200 milliseconds after receiving user data. The method described in Receiving Data (Section 3.6.3.2) sets the session's `DELACK_ALARM`.

When `DELACK_ALARM` fires: set `ACK_NOW` to true.

#### 3.6.3.4.5. Obligatory Acknowledgement

One or more acknowledgements should be sent as soon as is practical when the session's `ACK_NOW` flag is set. While the `ACK_NOW` flag is set:

1. Choose a receiving flow that is ready to send an acknowledgement;
2. If there is no such flow: there is no work to do, set `ACK_NOW` to false, set `RX_DATA_PACKETS` to 0, clear the `DELACK_ALARM`, and stop; otherwise
3. Start a new packet;

4. Assemble an acknowledgement for the flow and include it in the packet, truncating it if necessary so that the packet doesn't exceed the path MTU;
5. Set flow.SHOULD\_ACK to false;
6. Set flow.PREV\_RWND to the bufferBlocksAvailable field of the included acknowledgement chunk;
7. Attempt to piggyback acknowledgements for any other flows that are ready to send an acknowledgement into the packet, as described below; and
8. Send the packet.

#### 3.6.3.4.6. Opportunistic Acknowledgement

When sending a packet with user data or an acknowledgement, any other receiving flows that are ready to send an acknowledgement should include their acknowledgements in the packet if possible.

To piggyback acknowledgements in a packet that is already being sent, where the packet contains user data or an acknowledgement: While there is at least one receiving flow that is ready to send an acknowledgement:

1. Assemble an acknowledgement for the flow;
2. If the acknowledgement cannot be included in the packet without exceeding the path MTU: the packet is full, stop; otherwise
3. Include the acknowledgement in the packet;
4. Set flow.SHOULD\_ACK to false;
5. Set flow.PREV\_RWND to the bufferBlocksAvailable field of the included acknowledgement chunk; and
6. If there are no longer any receiving flows in the session that are ready to send an acknowledgement: set session.ACK\_NOW to false, set session.RX\_DATA\_PACKETS to 0, and clear session.DELACK\_ALARM.

#### 3.6.3.4.7. Example

Receiver

```

1 | <--- Data ID=3, seq#=29, fsnOff=11 (fsn=18)
2 | <--- Data ID=3, seq#=30, fsnOff=12 (fsn=18)
3 | ---> Ack ID=3, seq:0-30
4 | <--- Data ID=3, seq#=32, fsnOff=12 (fsn=20)
5 | ---> Ack ID=3, seq:0-30, 32
6 | <--- Data ID=3, seq#=34, fsnOff=12 (fsn=22)
7 | ---> Ack ID=3, seq:0-30, 32, 34
  | :
8 | <--- Data ID=3, seq#=46, fsnOff=16 (fsn=30)
9 | ---> Ack ID=3, seq:0-30, 32, 34-46
10 | <--- Data ID=3, seq#=47, fsnOff=15 (fsn=32)
11 | ---> Ack ID=3, seq:0-32, 34-47
12 | <--- Data ID=3, seq#=33, fsnOff=1 (fsn=32)
13 | ---> Ack ID=3, seq#=0-47
14 | <--- Data ID=3, seq#=48, fsnOff=16 (fsn=32)
15 | <--- Data ID=3, seq#=49, fsnOff=17 (fsn=32)
16 | ---> Ack ID=3, seq#=0-49
  | :

```

Flow with sequence numbers 31 and 33 lost in transit, 31 abandoned and 33 retransmitted.

Figure 23

### 3.6.3.5. Flow Control

The flow receiver maintains a buffer for reassembling and reordering messages for delivery to the user (Section 3.6.3.3). The implementation and the user may wish to limit the amount of resources (including buffer memory) that a flow is allowed to use.

RTMFP provides a means for each receiving flow to govern the amount of data sent by the sender, by way of the `bufferBytesAvailable` derived field of acknowledgement chunks (Section 2.3.13, Section 2.3.14). This derived field indicates the amount of data that the sender is allowed to have outstanding in the network, until instructed otherwise. This amount is also called the receive window.

The flow receiver can suspend the sender by advertising a closed (zero length) receive window.

The user can suspend delivery of messages from the receiving flow (Section 3.6.3.3). This can cause the receive buffer to fill.

In order for progress to be made on completing a fragmented message or repairing a gap for sequenced delivery in a flow, the flow receiver MUST advertise at least one buffer block in an



acknowledgement if it is not suspended, even if the amount of data in the buffer exceeds the buffer capacity, unless the buffer capacity is 0. Otherwise, deadlock can occur, as the receive buffer will stay full and won't drain because of a gap or incomplete message, and the gap or incomplete message can't be repaired or completed because the sender is suspended.

The receive window is advertised in units of 1024-byte blocks. For example, advertisements for 1 byte, 1023 bytes, and 1024 bytes each require one block. An advertisement for 1025 bytes requires two blocks.

The following describes the RECOMMENDED method of calculating the `bufferBlocksAvailable` field of an acknowledgement chunk for a receiving flow:

1. If `BUFFERED_SIZE` is greater than or equal to `BUFFER_CAPACITY`: set `ADVVERTISE_BYTES` to 0;
2. If `BUFFERED_SIZE` is less than `BUFFER_CAPACITY`: set `ADVVERTISE_BYTES` to `BUFFER_CAPACITY - BUFFERED_SIZE`;
3. Set `ADVVERTISE_BLOCKS` to `CEIL(ADVVERTISE_BYTES / 1024)`;
4. If `ADVVERTISE_BLOCKS` is 0, AND `BUFFER_CAPACITY` is greater than 0, AND delivery to the user is not suspended: set `ADVVERTISE_BLOCKS` to 1; and
5. Set the acknowledgement's `bufferBlocksAvailable` field to `ADVVERTISE_BLOCKS`.

#### 3.6.3.6. Receiving a Buffer Probe

A Buffer Probe chunk (Section 2.3.15) is sent by the flow sender (Section 3.6.2.9.1) to request the current receive window advertisement (in the form of an acknowledgement) from the flow receiver.

On receipt of a Buffer Probe chunk:

1. If `chunk.flowID` doesn't belong to a receiving flow in the same session in the `RF_OPEN`, `RF_REJECTED`, or `RF_COMPLETE_LINGER` state: ignore this Buffer Probe; otherwise,
2. Retrieve the receiving flow context for the flow indicated by `chunk.flowID`; then

3. Set flow.SHOULD\_ACK to true; and
4. Set session.ACK\_NOW to true.

#### 3.6.3.7. Rejecting a Flow

A receiver can reject an RF\_OPEN flow at any time and for any reason. To reject a receiving flow in the RF\_OPEN state:

1. Move to the RF\_REJECTED state;
2. Discard all entries in flow.RECV\_BUFFER, as they are no longer relevant;
3. If the user rejected the flow, set flow.EXCEPTION\_CODE to the exception code indicated by the user; otherwise the flow was rejected automatically by the implementation, so the exception code is 0;
4. Set flow.SHOULD\_ACK to true; and
5. Set session.ACK\_NOW to true.

The receiver indicates that it has rejected a flow by sending a Flow Exception Report chunk (Section 2.3.16) with every acknowledgement (Section 3.6.3.4.3) for a flow in the RF\_REJECTED state.

#### 3.6.3.8. Close

A receiving flow is complete when every sequence number from 0 through and including the final sequence number has been received; that is, when flow.RF\_FINAL\_SN has a value and flow.SEQUENCE\_SET contains every sequence number from 0 through flow.RF\_FINAL\_SN, inclusive.

When an RF\_OPEN or RF\_REJECTED receiving flow becomes complete, move to the RF\_COMPLETE\_LINGER state, set flow.SHOULD\_ACK to true, and set session.ACK\_NOW to true.

A receiving flow SHOULD remain in the RF\_COMPLETE\_LINGER state for 120 seconds. After 120 seconds, move to the RF\_CLOSED state. The receiving flow is now closed, and its resources can be reclaimed once all complete messages in flow.RECV\_BUFFER have been delivered to the user (Section 3.6.3.3). The same flow ID might be used for a new flow by the sender after this point.

Discussion: The flow sender detects that the flow is complete on receiving an acknowledgement of all fragment sequence numbers of the

flow. This can't happen until after the receiver has detected that the flow is complete and acknowledged all of the sequence numbers. The receiver's `RF_COMPLETE_LINGER` period is two minutes (one Maximum Segment Lifetime (MSL)), which allows any in flight packets to drain from the network without being misidentified, and gives the sender an opportunity to retransmit any sequence numbers if the completing acknowledgement is lost. The sender's `F_COMPLETE_LINGER` period is at least two minutes plus 10 seconds, and doesn't begin until the completing acknowledgement is received; therefore, the same flow identifier won't be re-used by the flow sender for a new sending flow for at least 10 seconds after the flow receiver has closed the receiving flow context. This ensures correct operation independent of network delay and even when the sender's clock runs up to 8 percent faster than the receiver's.

#### 4. IANA Considerations

This memo specifies chunk type code values (Section 2.3) and User Data option type code values (Section 2.3.11.1). These type code values are assigned and maintained by Adobe. Therefore, this memo has no IANA actions.

#### 5. Security Considerations

This memo specifies a general framework that can be used to establish a confidential and authenticated session between endpoints. A Cryptography Profile, not specified herein, defines the cryptographic algorithms, data formats, and semantics as used within this framework. Designing a Cryptography Profile to ensure that communications are protected to the degree required by the application-specific threat model is outside the scope of this specification.

A block cipher in CBC mode is RECOMMENDED for packet encryption (Section 2.2.3). An attacker can predict the values of some fields from one plain RTMFP packet to the next, or predict that some fields may be the same from one packet to the next. This SHOULD be considered in choosing and implementing a packet encryption cipher and mode.

The well-known Default Session Key of a Cryptography Profile serves multiple purposes, including: to scramble session startup packets to protect interior fields from undesirable modification by middleboxes such as NATs; to increase the effort required for casual passive observation of startup packets; to allow for different applications of RTMFP using different Default Session Keys to (intentionally or

not) share network transport addresses without interference. The Default Session Key, being well-known, MUST NOT be construed to contribute to the security of session startup; session startup is essentially in the clear.

Section 3.5.4.2 describes an OPTIONAL method for processing a change of network address of a communicating peer. Securely processing address mobility using that or any substantially similar method REQUIRES at least that the Packet Encryption function of the Cryptography Profile (Section 2.2.3) employs a cryptographic verification mechanism comprising secret information known only to the two endpoints. Without this constraint, that or any substantially similar method becomes "session hijacking support".

Flows and packet fragmentation imply semantics that could cause unbounded resource utilization in receivers, causing a denial of service. Implementations SHOULD guard against unbounded or excessive resource use, and abort sessions that appear abusive.

A rogue but popular Redirector (Section 3.5.1.4) could direct session Initiators to flood a victim address or network with Initiator Hello packets, potentially causing a denial of service.

An attacker that can passively observe an IHello and that possesses a certificate matching the Endpoint Discriminator (without having to know the private key, if any, associated with it) can deny the Initiator access to the desired Responder by sending an RHello before the desired Responder does, since only the first received RHello is selected by the Initiator. The attacker needn't forge the desired Responder's source address, since the RHello is selected based on the tag echo and not the packet's source address. This can simplify the attack in some network or host configurations.

An attacker that can passively observe and record the packets of an established session can use traffic analysis techniques to infer the start and completion of flows without decrypting the packets. The User Data fragments of flows have unique sequence numbers, so flows are immune to replay while they are open. However, once a flow has completed and the linger period has concluded, the attacker could replay the recorded packets, opening a new flow in the receiver and duplicating the flow's data, which might have undesirable effects in the receiver's application. The attacker could also infer that a new flow has begun reusing the recorded flow's identifier, and replay the final sequence number or any of the other fragments in the flow, potentially denying or interfering with legitimate traffic to the receiver. Therefore, the data integrity aspect of packet encryption SHOULD comprise anti-replay measures.

## 6. Acknowledgements

Special thanks go to Matthew Kaufman for his contributions to the creation and design of RTMFP.

Thanks to Jari Arkko, Ben Campbell, Wesley Eddy, Stephen Farrell, Philipp Hancke, Bela Lubkin, Hilarie Orman, Richard Scheffenegger, and Martin Stiemerling for their detailed reviews of this memo.

## 7. References

### 7.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.
- [CBC] Dworkin, M., "Recommendation for Block Cipher Modes of Operation", NIST Special Publication 800-38A, December 2001, <<http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>>.

### 7.2. Informative References

- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[ScalableTCP]

Kelly, T., "Scalable TCP: Improving Performance in Highspeed Wide Area Networks", December 2002, <<http://datatag.web.cern.ch/datatag/papers/pfldnet2003-ctk.pdf>>.

## Appendix A. Example Congestion Control Algorithm

Section 3.5.2 mandates that an RTMFP use TCP-compatible congestion control, but allows flexibility in exact implementation within certain limits. This section describes an experimental window-based congestion control algorithm that is appropriate for real-time and bulk data transport in RTMFP. The algorithm includes slow-start and congestion avoidance phases including modified increase and decrease parameters. These parameters are further adjusted according to whether real-time data is being sent and whether time-critical reverse notifications are received.

### A.1. Discussion

RFC 5681 defines the standard window-based congestion control algorithms for TCP. These algorithms are appropriate for delay-insensitive bulk data transport, but have undesirable behaviors for delay- and loss-sensitive applications. Among the undesirable behaviors are the cutting of the congestion window in half during a loss event, and the rapidity of the slow start algorithm's exponential growth. Cutting the congestion window in half requires a large channel headroom to support a real-time application, and can cause a large amount of jitter from sender-side buffering. Doubling the congestion window during the slow-start phase can lead to the congestion window temporarily growing to twice the size it should be, causing a period of excessive loss in the path.

We found that a number of deployed TCP implementations use the method of equation 3 from RFC 5681 Section 3.1, which, when combined with the recommended behavior of acknowledging every other packet, causes the congestion window to grow at approximately half the rate as the recommended method specifies. In order to compete fairly with these deployed TCPs, we choose 768 bytes per round trip as the increment during the normal congestion avoidance phase, which is approximately half of the typical maximum segment size of 1500 bytes while also being easily subdivided.

The sender may be sending real-time data to the far end. When sending real-time data, a smoother response to congestion is desired while still competing with reasonable fairness to other flows in the Internet. In order to scale the sending rate quickly, the slow start algorithm is desired, but slow start's normal rate of increase can

cause excessive loss in the last round trip. Accordingly, slow start's exponential increase rate is adjusted to double every approximately 3 round trips instead of every round trip. The multiplicative decrease cuts the congestion window by one eighth on loss to maintain a smoother sending rate. The additive increase is done at half the normal rate (incrementing at 384 bytes per round trip), both to compensate for the less aggressive loss response and to probe the path capacity more gently.

The far end may report that it is receiving real-time data from other peers, or the sender may be sending real-time data to other far ends. In these circumstances (if not sending real-time data to this far end) it is desirable to respond differently than the standard TCP algorithms specify, both to yield capacity to the real-time flows and to avoid excessive losses while probing the path capacity. Slow start's exponential increase is disabled and the additive increase is done at half the normal rate (incrementing at 384 bytes per round trip). Multiplicative decrease is left at cutting by half to yield to other flows.

Since real-time messages may be small, and sent regularly, it is advantageous to spread congestion window increases out across the round-trip time instead of doing them all at once. We divide the round-trip into 16 segments with an additive increase of a useful size (48 bytes) per segment.

Scalable TCP [ScalableTCP] describes experimental methods of modifying the additive increase and multiplicative decrease of the congestion window in large delay-bandwidth scenarios. The congestion window is increased by 1% each round trip and decreased by one eighth on loss in the congestion avoidance phase in certain circumstances (specifically, when a 1% increase is larger than the normal additive-increase amount). Those methods are adapted here. The scalable increase amount is 48 bytes for every 4800 bytes acknowledged, to spread the increase out over the round-trip. The congestion window is decreased by one eighth on loss when it is at least 67200 bytes per round trip, which is seven eighths of 76800 (the point at which 1% is greater than 768 bytes per round trip). When sending real-time data to the far end, the scalable increase is 1% or 384 bytes per round trip, whichever is greater. Otherwise, when notified that the far end is receiving real-time data from other peers, the scaled increase is adjusted to 0.5% or 384 bytes per round trip, whichever is greater.

## A.2. Algorithm

Let SMSS denote the Sender Maximum Segment Size [RFC5681], for example 1460 bytes. Let CWND\_INIT denote the Initial Congestion

Window (IW) according to RFC 5681 Section 3.1, for example 4380 bytes. Let CWND\_TIMEOUT denote the congestion window after a timeout indicating lost data, being 1\*SMSS (for example 1460 bytes).

Let the session information context contain additional variables:

- o CWND: the Congestion Window, initialized to CWND\_INIT;
- o SSTHRESH: the Slow Start Threshold, initialized to positive infinity;
- o ACKED\_BYTES\_ACCUMULATOR: a count of acknowledged bytes, initialized to 0;
- o ACKED\_BYTES\_THIS\_PACKET: a count of acknowledged bytes observed in the current packet;
- o PRE\_ACK\_OUTSTANDING: the number of bytes outstanding in the network before processing any acknowledgements in the current packet;
- o ANY\_LOSS: an indication to whether any loss has been detected in the current packet;
- o ANY\_NAKS: an indication to whether any negative acknowledgements have been detected in the current packet;
- o ANY\_ACKS: an indication to whether any acknowledgement chunks have been received in the current packet;

Let FASTGROW\_ALLOWED indicate whether the congestion window is allowed to grow at the normal rate versus a slower rate, being FALSE if a Time Critical Reverse Notification has been received on this session within the last 800 milliseconds (Section 2.2.4, Section 3.5.2.1) or if a Time Critical Forward Notification has been sent on ANY session in the last 800 milliseconds, and otherwise being TRUE.

Let TC\_SENT indicate whether a Time Critical Forward Notification has been sent on this session within the last 800 milliseconds.

Implement the method of Section 3.6.2.6 to manage transmission timeouts, including setting the TIMEOUT\_ALARM.

On being notified that the TIMEOUT\_ALARM has fired, perform the function in Figure 24:



```
on TimeoutNotification(WAS_LOSS):  
    set Ssthresh to MAX(Ssthresh, CWND * 3/4).  
    set ACKED_BYTES_ACCUMULATOR to 0.  
    if WAS_LOSS is TRUE:  
        set CWND to CWND_TIMEOUT.  
    else:  
        set CWND to CWND_INIT.
```

Figure 24: Pseudocode for handling a timeout notification

Before processing each received packet in this session:

1. Set ANY\_LOSS to FALSE;
2. Set ANY\_NAKS to FALSE;
3. Set ACKED\_BYTES\_THIS\_PACKET to 0; and
4. Set PRE\_ACK\_OUTSTANDING to S\_OUTSTANDING\_BYTES.

On notification of loss (Section 3.6.2.5): set ANY\_LOSS to TRUE.

On notification of negative acknowledgement (Section 3.6.2.5): set ANY\_NAKS to TRUE.

On notification of acknowledgement of data (Section 3.6.2.4): set ANY\_ACKS to TRUE, and add the count of acknowledged bytes to ACKED\_BYTES\_THIS\_PACKET.

After processing all chunks in each received packet for this session, perform the function in Figure 25:

```
if ANY_LOSS is TRUE:
    if (TC_SENT is TRUE) OR (PRE_ACK_OUTSTANDING > 67200 AND \
        FASTGROW_ALLOWED is TRUE):
        set Ssthresh to MAX(PRE_ACK_OUTSTANDING * 7/8, CWND_INIT).
    else:
        set Ssthresh to MAX(PRE_ACK_OUTSTANDING * 1/2, CWND_INIT).
    set CWND to Ssthresh.
    set ACKED_BYTES_ACCUMULATOR to 0.
else if (ANY_ACKS is TRUE) AND (ANY_NAKS is FALSE) AND \
    (PRE_ACK_OUTSTANDING >= CWND):
    set var INCREASE to 0.
    var Aithresh.
    if FASTGROW_ALLOWED is TRUE:
        if CWND < Ssthresh:
            set INCREASE to ACKED_BYTES_THIS_PACKET.
        else:
            add ACKED_BYTES_THIS_PACKET to ACKED_BYTES_ACCUMULATOR.
            set Aithresh to MIN(MAX(CWND / 16, 64), 4800).
            while ACKED_BYTES_ACCUMULATOR >= Aithresh:
                subtract Aithresh from ACKED_BYTES_ACCUMULATOR.
                add 48 to INCREASE.
    else FASTGROW_ALLOWED is FALSE:
        if CWND < Ssthresh AND TC_SENT is TRUE:
            set INCREASE to CEIL(ACKED_BYTES_THIS_PACKET / 4).
        else:
            var Aithresh_CAP.
            if TC_SENT is TRUE:
                set Aithresh_CAP to 2400.
            else:
                set Aithresh_CAP to 4800.
            add ACKED_BYTES_THIS_PACKET to ACKED_BYTES_ACCUMULATOR.
            set Aithresh to MIN(MAX(CWND / 16, 64), Aithresh_CAP).
            while ACKED_BYTES_ACCUMULATOR >= Aithresh:
                subtract Aithresh from ACKED_BYTES_ACCUMULATOR.
                add 24 to INCREASE.
    set CWND to MAX(CWND + MIN(INCREASE, SMSS), CWND_INIT).
```

Figure 25: Pseudocode for congestion window adjustment after processing a packet

Author's Address

Michael C. Thornburgh  
Adobe Systems Incorporated  
345 Park Avenue  
San Jose, CA 95110-2704  
US

Phone: +1 408 536 6000  
Email: [mthornbu@adobe.com](mailto:mthornbu@adobe.com)  
URI: <http://www.adobe.com/>



Network Working Group  
Internet Draft  
Category: Standard Track

L. Yong  
X. Xu  
Huawei

Expires: August 2013

February 25, 2013

Generic UDP Encapsulation for IP Tunneling  
draft-yong-tsvwg-udp-encap-4-ip-tunneling-01

Status of this document

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/ietf/lid-abstracts.txt>.

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

This Internet-Draft will expire on August 25, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Abstract

This document describes a method for encapsulating the layer protocols within UDP at an IP network edge such that the payload protocol can be identified from the destination port value. The mechanism also allows for the source port to be used as the entropy field for the purpose of load balancing in environments such as Equal Cost Multipath (ECMP) in the underlying IP network.

Application of this technique is focused on tunneling payload network flows across IP networks in environments where load-balancing is required. No changes to the underlying IP network or the payload networks are required.

## Table of Contents

|                                             |    |
|---------------------------------------------|----|
| 1. Introduction.....                        | 3  |
| 1.1. Conventions used in this document..... | 4  |
| 1.2. Terminology.....                       | 4  |
| 2. UDP Encapsulation.....                   | 5  |
| 3. Procedures.....                          | 5  |
| 4. Encapsulation Considerations.....        | 6  |
| 5. Backward Compatibility.....              | 7  |
| 6. IP Tunneling Applications.....           | 7  |
| 7. Security Considerations.....             | 8  |
| 8. IANA Considerations.....                 | 8  |
| 9. Contributors.....                        | 11 |
| 10. Acknowledgements.....                   | 12 |
| 11. References.....                         | 12 |
| 11.1. Normative References.....             | 12 |
| 11.2. Informative References.....           | 13 |

## 1. Introduction

Load balancing is an attempt to balance traffic across a network by allowing the traffic to use multiple paths. The benefits of load balancing are: it eases capacity planning; it can help absorb traffic surges by spreading them across multiple paths; and it allows better resilience by offering alternate paths in the event of a link or node failure.

Load balancing of traffic over Equal Cost Multi-Path (ECMP) and/or Link Aggregation Group (LAG) in IP networks is widely used. Most existing routers in IP networks are already capable of distributing IP traffic flows over ECMP paths and/or LAG on the basis of a hash function performed on the key fields of 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) packet, the hash operates on the five-tuple of the source IP address, the destination IP address, the source port, the destination port, and the next protocol identifier.

IP Tunneling is a technique for allowing a tunneled packet (IP or non-IP) to transit an IP network by encapsulating it within an IP header when it enters the IP network and decapsulating it when it leaves the IP network. Several IP tunneling techniques exist for an IP network to support an IP tunneling application. IP-in-IP [RFC5565] carries IP encapsulated directly in another IP header. Generic Routing Encapsulation (GRE) [RFC2784] provides an encapsulation header to allow any protocol to be carried in an IP tunnel. [RFC4023] describes how to carry MPLS packets in IP or GRE headers. Version 3 of the Layer Two Tunneling Protocol (L2TPv3) [RFC3931] defines a control protocol and encapsulation for carrying multiple layer 2 connections between two IP nodes.

In order to leverage the benefits of multipath environments, it is necessary to ensure that traffic flows are not distributed across multiple paths. When a single IP flow may carry a number of payload traffic flows, this requires that the payload flows can be identified in a way that the hash function will make the right decisions. An example of the way this works can be seen in [RFC6790] that introduced the MPLS Entropy Label, which allows information about the traffic flow with which a given packet is associated to be encoded within a special label within the MPLS label stack and form part of the hash that an MPLS transit node uses when distributing flows over multiple paths.

IP in IP is able to encode the payload type, but cannot easily carry the information necessary to achieve load balancing. GRE is able to encode the payload type and carry load balancing information; however, special processing at transit routers is required to understand the load balancing information because the GRE header does not form part of the standard hash function. L2TPv3 has the same capabilities and problems as GRE. Thus, none of the existing mechanisms for tunneling the layer protocols supported at network edge across an IP network provides a way to make good use of multipath environments.

This document defines a generic UDP-based encapsulation for tunneling the layer protocols across IP networks. The encapsulation encodes the payload type in the UDP destination port and uses the UDP source port to provide load balancing information in a way that mirrors the entropy label of [RFC6790]. This encapsulation requires no changes to the transit IP network nor to the networks whose traffic is transiting the IP network. In particular, hash functions in existing IP routers will automatically utilize and benefit from this procedure without needing any change or upgrade. The encapsulation mechanism is applicable to a variety of IP networks including Data Centers and Wide Area infrastructure networks.

Note that [RFC5405] provides unicast UDP usage guidelines for application designers. The application in that context is about the unicast application above IP network layer and the design considerations are for the upper layer application when it uses the UDP as transport protocol. This document proposes the use of the UDP header to encode the packet type and load balancing information for the IP network in environments of ECMP. In other words, the UDP usage here is not to facilitate the transport of upper layer application data. Therefore RFC5405 design considerations do not apply to the case described in this document.

#### 1.1. 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].

#### 1.2. Terminology

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



## 2. UDP Encapsulation

The UDP datagram format is described in [RFC768]. The encapsulation described in this document specifies that:

- . The destination port of the UDP datagram is set to indicate the payload protocol according to the values assigned by IANA as described in Section 8;
- . The source port may be set to any value by the sender. Varying the value of the source port according to the payload flow will enable load balancing within the IP network.
- . Other UDP datagram header fields are to be used as described in [RFC768].
- . To simplify the operation at the tunnel egress, it is RECOMMENDED that the UDP checksum field is set to zero. The encapsulation can apply to both IPv4 [RFC791] and IPv6 [RFC2460] tunnel. For further considerations related to relaxation of the mandatory requirement for the use of a UDP checksum in an IPv6 network, please refer to [CKZERO] and [CKSUM].

## 3. Procedures

When a tunnel ingress conforming to this document receives a packet, the ingress MUST encapsulate the packet into a UDP packet as described in Section 2. The ingress MUST insert the payload type in the destination port field. The ingress MAY generate load balancing information and put it in the source port field, otherwise it SHOULD be set to zero. How a tunnel ingress generates the load balancing information from the payload is outside the scope of this document. The tunnel ingress MUST encode its IP address as the source IP address and the egress tunnel endpoint IP address or a group IP address as 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 tunnel egress endpoint.

Transit routers, upon receiving these UDP encapsulated packets, may load-balance these packets based on the hash of the five-tuple of the packet header. Note that this method requires no change on transit routers.

When the tunnel egress receives a packet, it removes the UDP header and sends the payload to the entity that will process the payload

type indicated in the UDP destination port. Section 5 describes the error handling when this entity is not instantiated at the tunnel egress. When a packet is received with a UDP checksum of zero, it MUST be accepted for decapsulation. Although IPv6 [RFC2460] restricts the processing a packet with the UDP checksum of zero, [CKSUM] and [CKZERO] relax this constraint to allow the zero UDP checksum. Note that 1) the IPv6 tunnel ingress and egress SHOULD follow the node requirements specified in Section 4 of [CKZERO] and the usage requirements specified in Section 5 of [CKZERO]; 2) IPv6 transit nodes SHOULD follow the requirements 9, 10, 11 specified in Section 5 of [CKZERO].

Note that the UDP encapsulation specified in this document does not provide a flow control capability; it is the responsibility of tunneled network protocol to support any necessary flow control function.

#### 4. Encapsulation Considerations

This UDP encapsulation allows the tunneled traffic to be unicast, broadcast, or multicast traffic. The load balancing information may be generated from the header of tunneled unicast or broadcast/multicast packets at a tunnel ingress. The mapping mechanism between the tunneled multicast traffic and the multicast capability in the IP network is transparent and independent to the encapsulation and is outside the scope of this document.

UDP encapsulation introduces eight octets overhead, which reduces the effective Maximum Transmission Unit (MTU) size. If a tunnel ingress has to perform fragmentation on a packet before encapsulation, it MUST use the same source UDP port for all packet fragments. This ensures that the transit routers will forward the packet fragments on the same path. An operator should factor in this addition eight octets when considering an MTU size for the payload to reduce the likelihood of fragmentation.

To ensure the tunneled traffic gets the same treatment over the IP network, prior to the encapsulation process, a tunnel ingress needs process the payload to get the proper parameters to fill into the IP header such as DiffServ [RFC2983]. Both tunnel ingress and egress SHOULD follow the procedures described in RFC6040 [RFC6040] for ECN

marking propagation. This process is outside of the scope of this document.

Note that IPv6 header [RFC2460] has a flow label field that may be used for load balancing information in the IPv6 network [RFC6438]. The next header in IPv6 can be used to provide the payload type. However, applying the UDP encapsulation to both IPv4 and IPv6 networks provides a unified hardware implementation for load balancing in an IP network.

## 5. Backward Compatibility

It is assumed that tunnel ingress routers are upgraded in order to support the function described in this document.

No change is required at transit routers to support the process described in this document.

If a legacy router that is an intended tunnel egress does not support the UDP encapsulation described in this document, it will not be listening on the destination port. The same will apply if the egress supports the process but not the payload protocol. In these cases, the router will conform to normal UDP processing and respond to the tunnel ingress with an ICMP message indicating "port unreachable" according to [RFC792] and [RFC4884]. Upon receiving this ICMP message, the tunnel ingress MUST NOT continue to use the UDP encapsulation toward this tunnel egress without management intervention.

## 6. IP Tunneling Applications

IP tunneling applications such as IP in IP [RFC5565], MPLS Client Layer [EVPN] and L3VPN [RFC4364], GRE [RFC2784], VXLAN [VXLAN], NVGRE [NVGRE], LISP [RFC6830] may be deployed in an IP network. These applications can use the UDP encapsulation described in this document. In fact, VXLAN and LISP are specified to use the UDP header and have exactly the same semantics as specified in this document.

The UDP encapsulation uses the destination port to encode the payload type. For data plane interworking, each application MUST have one port number assigned by IANA. An application MAY request more than one port number [MPLS-IN-UDP]. Each application MUST further specify its header format and semantics if not yet. For a

network layer virtualization application, its header needs to have a field to sufficiently identify a virtual network. The header format of such application is outside the scope of this document.

Some tunneling applications may use of a signal protocol to exchange information between two tunnel end points at the application layer. Such signal protocol can be used for egress to inform its capability in supporting the UDP encapsulation as well. Again, this is outside the scope of this document.

## 7. Security Considerations

UDP encapsulation does not provide any additional security for a tunneled layer protocol. In other words, the security concern has no change regarding the use of UDP encapsulation or not. An IP tunneling application either relies on the underlying IP network to provide the security or implement a secured overlay tunnel such as L2TPv3 [RFC3931] to over a non-secured IP network such as Internet.

Note that the UDP header usage described in this document is for the underlying IP network. A firewall in an underlying network SHOULD not inspect the packets and take an action because of the UDP header presence for this purpose.

## 8. IANA Considerations

This document requests IANA to reserve a block of port numbers from the range designated as User Ports for the payload types that the IP tunnel may carry. IANA is requested to manage that block as a sub-registry of the port numbers registry.

Note that the use of a contiguous block of port numbers is not an absolute requirement, however it is believed that the use of such a block will considerably aid implementation.

Allocation procedures for port numbers are governed by [RFC6335]. This document does not seek to change those procedures. Instead, it requests that a contiguous block of port numbers be assigned as below, and that a further contiguous block be considered reserved for allocation for similar purposes.

IANA is requested to make the following allocations:

Service Name : IPv4-UDP-ECMP  
Transport Protocol(s) : UDP  
Assignee : IESG <iesg@ietf.org>  
Contact : IETF Chair <chair@ietf.org>  
Description : IPv4 encapsulation in UDP for load balancing.  
Reference : [This.I-D]  
Port Number : TBD  
Service Code : N/A  
Known Unauthorized Uses : N/A  
Assignment Notes : N/A

Service Name : IPv6-UDP-ECMP  
Transport Protocol(s) : UDP  
Assignee : IESG <iesg@ietf.org>  
Contact : IETF Chair <chair@ietf.org>  
Description : IPv6 encapsulation in UDP for load balancing.  
Reference : [This.I-D]  
Port Number : TBD + 1  
Service Code : N/A  
Known Unauthorized Uses : N/A  
Assignment Notes : N/A

Service Name : MPLS-up-UDP-ECMP  
Transport Protocol(s) : UDP  
Assignee : IESG <iesg@ietf.org>  
Contact : IETF Chair <chair@ietf.org>  
Description : MPLS upstream assigned label encapsulation in  
UDP for load balancing.  
Reference : [This.I-D]  
Port Number : TBD + 2  
Service Code : N/A  
Known Unauthorized Uses : N/A  
Assignment Notes : N/A

Service Name : MPLS-dn-UDP-ECMP  
Transport Protocol(s) : UDP  
Assignee : IESG <iesg@ietf.org>  
Contact : IETF Chair <chair@ietf.org>  
Description : MPLS downstream assigned label encapsulation in  
UDP for load balancing.  
Reference : [This.I-D]  
Port Number : TBD + 3

Service Code : N/A  
Known Unauthorized Uses : N/A  
Assignment Notes : N/A

Service Name : VXLAN-UDP-ECMP  
Transport Protocol(s) : UDP  
Assignee : IESG <iesg@ietf.org>  
Contact : IETF Chair <chair@ietf.org>  
Description : VXLAN encapsulation in UDP for load balancing.  
Reference : [This.I-D]  
Port Number : TBD + 4  
Service Code : N/A  
Known Unauthorized Uses : N/A  
Assignment Notes : N/A

Service Name : NVGRE-UDP-ECMP  
Transport Protocol(s) : UDP  
Assignee : IESG <iesg@ietf.org>  
Contact : IETF Chair <chair@ietf.org>  
Description : NVGRE encapsulation in UDP for load balancing.  
Reference : [This.I-D]  
Port Number : TBD + 5  
Service Code : N/A  
Known Unauthorized Uses : N/A  
Assignment Notes : N/A

Service Name : GRE-UDP-ECMP  
Transport Protocol(s) : UDP  
Assignee : IESG <iesg@ietf.org>  
Contact : IETF Chair <chair@ietf.org>  
Description : GRE encapsulation in UDP for load balancing.  
Reference : [This.I-D]  
Port Number : TBD + 6  
Service Code : N/A  
Known Unauthorized Uses : N/A  
Assignment Notes : N/A

IANA is requested to reserve a further nine (9) adjacent port numbers for similar uses. Any future allocation from that reserved block must follow the procedures defined in [RFC6335] and should:

- Be for the purpose of tunneling a layer protocol across a multipath-enabled IP network.
- Be the result of a Standards Track RFC
- Use the service name naming convention of <foo>-UDP-ECMP where <foo> is an identifier of the tunneled protocol.

## 9. Contributors

Edward Crabbe  
Google, Inc.  
1600 Amphitheatre Parkway  
Mountain View, CA 94043

Email: edc@google.com

John E. Drake  
Juniper Networks

Email: jdrake@juniper.net

Adrian Farrel  
Juniper Networks

Email: adrian@olddog.co.uk

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

Email: vishwas.manral@hp.com

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

Email: cpignata@cisco.com

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

Email: fanyb@gsta.com

Yiu Lee  
Comcast  
One Comcast Center  
Philadelphia, PA 1903  
USA

Email: Yiu\_Lee@Cable.Comcast.com

## 10. Acknowledgements

Authors like to thank Vivek Kumar, Ron Bonica, Joe Touch, Ruediger Geib, and Gorry Fairhurst for their review and valuable input on this draft.

## 11. References

### 11.1. Normative References

- [RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC791] DARPA, "Internet Protocol", RFC791, September 1981
- [RFC792] Postel, J., "Internet Control Message Protocol", RFC792, September, 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC2119, March 1997.
- [RFC2460] Deering, S., and Hinden, R., "Internet Protocol, Version 6 (IPv6)", RFC2460, December 1998
- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.
- [RFC3931] Lau, J., Townsley, M., and I. Goyret, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, March 2005.
- [RFC4023] Worster, T., Rekhter, Y., and E. Rosen, "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, March 2005.



- [RFC4364] Rosen, E and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC4364, February 2006.
- [RFC4884] Conta, A, et al., "Internet Control Message Protocol (ICMP) for the Internet Protocol Version 6 (IPv6) Specification", RFC4884, March, 2006
- [RFC5405] Eggert, L., "Unicast UDP Usage Guideline for Application Designers", RFC5405, November 2008.
- [RFC5565] Wu, J., "Softwire Mesh Framework", RFC 5565, June 2009.
- [RFC6040] Briscoe, B., "Tunneling of Explicit Congestion Notification", RFC6040, November 2010
- [RFC6335] Cotton, M., etc, "Internet Assigned Numbers Authority IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", RFC6335, August 2011
- [RFC6438] Carpenter, B., Amante, S., "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in tunnels", RFC6438, November, 2011
- [RFC6790] Kompella, K., et al. "The use of Entropy Label in MPLS forwarding", RFC6790, November 2012

#### 11.2. Informative References

- [CKSUM] Eubanks, M., et al., "IPv6 and UDP Checksums for Tunneled Packets", draft-ietf-6man-udpchecksums, work in progress.
- [CKZERO] G. Fairhurst, G. and Westerlund, M., "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums", draft-ietf-6man-udpzero, work in progress.
- [EVPN] Sajassi, A., et al., "BGP MPLS based Ethernet VPN", draft-ietf-l2vpn-evpn, work in progress.
- [MPLS-IN-UDP] Xu, X., et al., "Encapsulating MPLS in UDP", draft-ietf-mpls-in-udp, work in progress.
- [NVGRE] Sridharan, M., "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre, work in progress.

[RFC2983] Black, D. "Diffserv and tunnels", RFC2983, October 2000

[RFC6830] Farinacci, D., "Locator/ID Separation Protocol", RFC6830, January 2013.

[VXLAN] Mahalingam, M., Dutt, D., et al., "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", draft-mahalingam-dutt-dcops-vxlan, work in progress.

#### Authors' Addresses

Lucy Yong  
Huawei USA  
5340 Legacy Drive  
Plano, TX 75025  
U.S.A

Phone: 469-277-5837  
Email: lucy.yong@huawei.com

Xiaohu Xu  
Huawei Technologies,  
Beijing, China

Phone: +86-10-60610041  
Email: xuxiaohu@huawei.com

