SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2019

Z. Ali
C. Filsfils
N. Kumar
C. Pignataro
F. Iqbal
R. Gandhi
Cisco Systems, Inc.
J. Leddy
Comcast
S. Matsushima
SoftBank
R. Raszuk
Bloomberg LP
D. Voyer
Bell Canada
G. Dawra
LinkedIn
B. Peirens
Proximus
M. Chen
Huawei
G. Naik
Drexel University
October 22, 2018

Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)
draft-ali-spring-srv6-oam-02.txt

Abstract

This document defines building blocks that can be used for Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Dataplane (SRv6). The document also describes some SRv6 OAM mechanisms that can be realized using these building blocks.

Table of Contents

1. Introduction

This document defines building blocks that can be used for Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Dataplane (SRv6). The document also describes some SRv6 OAM mechanisms that can be implemented using these building blocks.

Additional OAM mechanisms will be added in a future revision of the document.

2. Conventions Used in This Document

2.1. Abbreviations

ECMP: Equal Cost Multi-Path.

SID: Segment ID.

SL: Segment Left.

SR: Segment Routing.

SRH: Segment Routing Header.

SRv6: Segment Routing with IPv6 Data plane.

TC: Traffic Class.

UCMP: Unequal Cost Multi-Path.

2.2. Terminology and Reference Topology

This document uses the terminology defined in [I-D.draft-filsfils-spring-srv6-network-programming]. The readers are expected to be familiar with the same.

Throughout the document, the following simple topology is used for illustration.

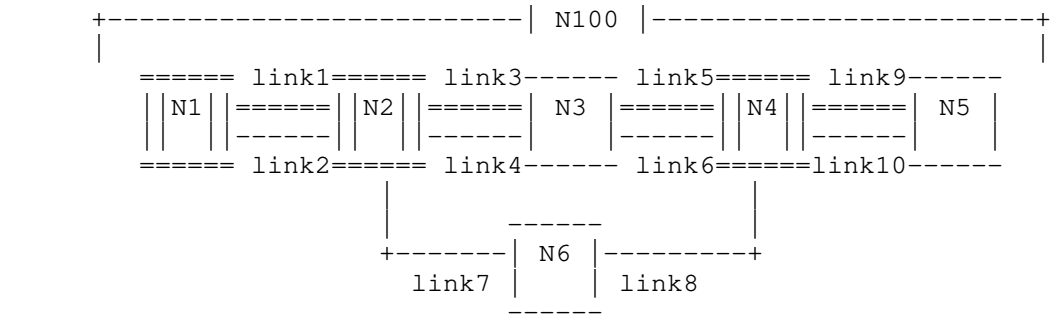


Figure 1 Reference Topology

In the reference topology:

Nodes N1, N2, and N4 are SRv6 capable nodes.

Nodes N3, N5 and N6 are classic IPv6 nodes.

Node N100 is a controller.

Node k has a classic IPv6 loopback address A:k::/128.
A SID at node k with locator block B and function F is represented by B:k:F::

The IPv6 address of the nth Link between node X and Y at the X side is represented as 2001:DB8:X:Y:Xn::, e.g., the IPv6 address of link6 (the 2nd link) between N3 and N4 at N3 in Figure 1 is 2001:DB8:3:4:32::. Similarly, the IPv6 address of link5 (the 1st link between N3 and N4) at node 3 is 2001:DB8:3:4:31::.

B:k:1:: is explicitly allocated as the END function at Node k.

B:k::Cij is explicitly allocated as the END.X function at node k towards neighbor node i via jth Link between node i and node j. e.g., B:2:C31 represents END.X at N2 towards N3 via link3 (the 1st link between N2 and N3). Similarly, B:4:C52 represents the END.X at N4 towards N5 via link10.

<S1, S2, S3> represents a SID list where S1 is the first SID and S3 is the last SID. (S3, S2, S1; SL) represents the same SID list but encoded in the SRH format where the rightmost SID (S1) in the SRH is the first SID and the leftmost SID (S3) in the SRH is the last SID.

(SA, DA) (S3, S2, S1; SL) represents an IPv6 packet, SA is the IPv6 Source Address, DA the IPv6 Destination Address, (S3, S2, S1; SL) is the SRH header that includes the SID list <S1, S2, S3>.

3. OAM Building Blocks

This section defines the various building blocks that can be used to implement OAM mechanisms in SRv6 networks. The following section describes some SRv6 OAM mechanisms that can be implemented using these building blocks.

3.1. O-flag in Segment Routing Header

[I-D. draft-ietf-6man-segment-routing-header] describes the Segment Routing Header (SRH) and how SR capable nodes use it. The draft [I-D. draft-ietf-6man-segment-routing-header] also define an OAM flag (SRH.Flags.O), which indicates that this packet is an operations and management (OAM) packet. The SRH draft also defines the processing rules for the O-flag in the SRH.Flags. The O-flag is one of the OAM building blocks considered in this document.

3.2. OAM Segments

OAM Segment IDs (SIDs) is another components of the building blocks needed to implement SRv6 OAM mechanisms. This document defines a couple of OAM SIDs. Additional SIDs will be added in the later version of the document.

3.2.1. End.OP: OAM Endpoint with Punt

Many scenarios require punting of SRv6 OAM packets at the desired nodes in the network. The "OAM Endpoint with Punt" function (End.OP for short) represents a particular OAM function to implement the punt behavior for an OAM packet. It is described using the pseudocode as follows:

When N receives a packet destined to S and S is a local End.OP SID, N does:

1. Punt the packet to CPU for SW processing (slow-path) ;; Ref1

Ref1: Hardware (microcode) only punts the packet. There is no requirement for the hardware to manipulate any TLV in the SRH (or elsewhere). Software (slow path) implements the required OAM mechanisms.

Please note that in an SRH containing END.OP SID, it is RECOMMENDED to set the SRH.Flags.O-flag = 0.

3.2.2. End.OTP: OAM Endpoint with Timestamp and Punt

Scenarios demanding performance management of an SR policy/ path requires hardware timestamping before hardware punts the packet to the software for OAM processing. The "OAM Endpoint with Timestamp and Punt" function (End.OTP for short) represents an OAM SID function to implement the timestamp and punt behavior for an OAM packet. It is described using the pseudocode as follows:

When N receives a packet destined to S and S is a local End.OTP SID, N does:

1. Timestamp the packet ;; Ref1

2. Punt the packet to CPU for SW processing (slow-path) ;; Ref2

Ref1: Timestamping is done in hardware, as soon as possible during the packet processing.

Ref2: Hardware (microcode) only punts the packet. There is no requirement for the hardware to manipulate any TLV in the SRH (or elsewhere). Software (slow path) implements the required OAM mechanisms.

Please note that in an SRH containing END.OTP SID, it is RECOMMENDED to set the SRH.Flags.O-flag = 0.

4. OAM Mechanisms

This section describes how OAM mechanisms can be implemented using the OAM building blocks described in the previous section. Additional OAM mechanisms will be added in a future revision of the document.

[RFC4443] describes Internet Control Message Protocol for IPv6 (ICMPv6) that is used by IPv6 devices for network diagnostic and error reporting purposes. As Segment Routing with IPv6 data plane (SRv6) simply adds a new type of Routing Extension Header, existing ICMPv6 ping mechanisms can be used in an SRv6 network. This section describes the applicability of ICMPv6 in the SRv6 network and how the existing ICMPv6 mechanisms can be used for providing OAM functionality.

Throughout this document, unless otherwise specified, the acronym ICMPv6 refers to multi-part ICMPv6 messages [RFC4884]. The document does not propose any changes to the standard ICMPv6 [RFC4443], [RFC4884] or standard ICMPv4 [RFC792].

4.1. Ping

There is no hardware or software change required for ping operation at the classic IPv6 nodes in an SRv6 network. That includes the classic IPv6 node with ingress, egress or transit roles. Furthermore, no protocol changes are required to the standard ICMPv6 [RFC4443], [RFC4884] or standard ICMPv4 [RFC792]. In other words, existing ICMP ping mechanisms work seamlessly in the SRv6 networks.

The following subsections outline some use cases of the ICMP ping in the SRv6 networks.

4.1.1. Classic Ping

The existing mechanism to ping a remote IP prefix, along the shortest path, continues to work without any modification. The initiator may be an SRv6 node or a classic IPv6 node. Similarly, the egress or transit may be an SRv6 capable node or a classic IPv6 node.

If an SRv6 capable ingress node wants to ping an IPv6 prefix via an arbitrary segment list <S1, S2, S3>, it needs to initiate ICMPv6 ping with an SR header containing the SID list <S1, S2, S3>. This is illustrated using the topology in Figure 1. Assume all the links have IGP metric 10 except both links between node2 and node3, which have IGP metric set to 100. User issues a ping from node N1 to a loopback of node 5, via segment list <B:2:C31, B:4:C52>.

Figure 2 contains sample output for a ping request initiated at node N1 to the loopback address of node N5 via a segment list <B:2:C31, B:4:C52>.

```
> ping A:5:: via segment-list B:2:C31, B:4:C52

Sending 5, 100-byte ICMP Echos to B5::, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 0.625
/0.749/0.931 ms
```

Figure 2 A sample ping output at an SRv6 capable node

All transit nodes process the echo request message like any other data packet carrying SR header and hence do not require any change. Similarly, the egress node (IPv6 classic or SRv6 capable) does not require any change to process the ICMPv6 echo request. For example, in the ping example of Figure 2:

- Node N1 initiates an ICMPv6 ping packet with SRH as follows (A:1::, B:2:C31)(A:5::, B:4:C52, B:2:C31, SL=2, NH = ICMPv6)(ICMPv6 Echo Request).
- Node N2, which is an SRv6 capable node, performs the standard SRH processing. Specifically, it executes the END.X function (B:2:C31) on the echo request packet.
- Node N3, which is a classic IPv6 node, performs the standard IPv6 processing. Specifically, it forwards the echo request based on DA B:4:C52 in the IPv6 header.
- Node N4, which is an SRv6 capable node, performs the standard SRH processing. Specifically, it observes the END.X function (B:4:C52) with PSP (Penultimate Segment POP) on the echo request packet and removes the SRH and forwards the packet across link10 to N5.
- The echo request packet at N5 arrives as an IPv6 packet without a SRH. Node N5, which is a classic IPv6 node, performs the standard IPv6/ ICMPv6 processing on the echo request and responds, accordingly.

4.1.2. Pinging a SID Function

The classic ping described in the previous section cannot be used to ping a remote SID function, as explained using an example in the following.

Consider the case where the user wants to ping the remote SID function B:4:C52, via B:2:C31, from node N1. Node N1 constructs the ping packet (A:1::, B:2:C31)(B:4:C52, B:2:C31, SL=1; NH=ICMPv6)(ICMPv6 Echo Request). The ping fails because the node N4 receives the ICMPv6 echo request with DA set to B:4:C52 but the next header is ICMPv6, instead of SRH. To solve this problem, the initiator needs to mark the ICMPv6 echo request as an OAM packet.

The OAM packets are identified either by setting the O-flag in SRH or by inserting the END.OP/ END.OTP SIDs at an appropriate place in the SRH. The following illustration uses END.OTP SID but the procedures are equally applicable to the END.OP SID.

In an SRv6 network, the user can exercise two flavors of the ping: end-to-end ping or segment-by-segment ping, as outlined in the following.

4.1.2.1. End-to-end ping using END.OP/ END.OTP

The end-to-end ping illustration uses the END.OTP SID but the procedures are equally applicable to the END.OP SID.

Consider the same example where the user wants to ping a remote SID function B:4:C52 , via B:2:C31, from node N1. To force a punt of the ICMPv6 echo request at the node N4, node N1 inserts the END.OTP SID just before the target SID B:4:C52 in the SRH. The ICMPv6 echo request is processed at the individual nodes along the path as follows:

- Node N1 initiates an ICMPv6 ping packet with SRH as follows (A:1::, B:2:C31)(B:4:C52, B:4:OTP, B:2:C31; SL=2; NH=ICMPv6)(ICMPv6 Echo Request).
- Node N2, which is an SRv6 capable node, performs the standard SRH processing. Specifically, it executes the END.X function (B:2:C31) on the echo request packet.
- Node N3 receives the packet as follows (A:1::, B:4:OTP)(B:4:C52, B:4:OTP, B:2:C31 ; SL=1; NH=ICMPv6)(ICMPv6 Echo Request). Node N3, which is a classic IPv6 node, performs the standard IPv6 processing. Specifically, it forwards the echo request based on DA B:4:OTP in the IPv6 header.
- When node N4 receives the packet (A:1::, B:4:OTP)(B:4:C52, B:4:OTP, B:2:C31 ; SL=1; NH=ICMPv6)(ICMPv6 Echo Request), it processes the END.OTP SID, as described in the pseudocode in Section 3. The packet gets punted to the ICMPv6 process for processing. The ICMPv6 process checks if the next SID in SRH (the target SID B:4:C52) is locally programmed.
- If the target SID is not locally programmed, N4 responses with the ICMPv6 message (Type: "SRv6 OAM (TBA)", Code: "SID not locally implemented (TBA)"); otherwise a success is returned.

4.1.2.2. Segment-by-segment ping using O-flag (Proof of Transit)

Consider the same example where the user wants to ping a remote SID function B:4:C52, via B:2:C31, from node N1. However, in this ping, the node N1 wants to get a response from each segment node in the SRH as a "proof of transit". In other words, in the segment-by-segment ping case, the node N1 expects a response from node N2 and node N4 for their respective local SID function. When a response to O-bit is desired from the last SID in a SID-list, it is the responsibility of the ingress node to use USP as the last SID. E.g., in this example, the target SID B:4:C52 is a USP SID.

To force a punt of the ICMPv6 echo request at node N2 and node N4, node N1 sets the O-flag in SRH. The ICMPv6 echo request is processed at the individual nodes along the path as follows: and

- Node N1 initiates an ICMPv6 ping packet with SRH as follows (A:1::, B:2:C31)(B:4:C52, B:2:C31; SL=1, Flags.O=1; NH=ICMPv6)(ICMPv6 Echo Request).
- When node N2 receives the packet (A:1::, B:2:C31)(B:4:C52, B:2:C31; SL=1, Flags.O=1; NH=ICMPv6)(ICMPv6 Echo Request) packet, it processes the O-flag in SRH, as described in the pseudocode in Section 3. A time-stamped copy of the packet gets punted to the ICMPv6 process for processing. Node N2 continues to apply the B:2:C31 SID function on the original packet and forwards it, accordingly. As B:4:C52 is a USP SID, N2 does not remove the SRH.
  The ICMPv6 process at node N2 checks if its local SID (B:2:C31) is locally programmed or not and responds to the ICMPv6 Echo Request.
- If the target SID is not locally programmed, N4 responses with the ICMPv6 message (Type: "SRv6 OAM (TBA)", Code: "SID not locally implemented (TBA)"); otherwise a success is returned. Please note that, as mentioned in Section 3, if node N2 does not support the O-flag, it simply ignores it and process the local SID, B:2:C31.
- Node N3, which is a classic IPv6 node, performs the standard IPv6 processing. Specifically, it forwards the echo request based on DA B:4:C52 in the IPv6 header.
- When node N4 receives the packet (A:1::, B:4:C52)(B:4:C52, B:2:C31; SL=0, Flags.O=1; NH=ICMPv6)(ICMPv6 Echo Request), it processes the O-flag in SRH, as described in the pseudocode in Section 3. A time-stamped copy of the packet gets punted to the ICMPv6 process for processing. The ICMPv6 process at node N4 checks if its local SID (B:2:C31) is locally programmed or not and responds to the ICMPv6 Echo Request. If the target SID is not locally programmed, N4 responses with the ICMPv6 message (Type: "SRv6 OAM (TBA)", Code: "SID not locally implemented (TBA)"); otherwise a success is returned.

Support for O-flag is part of node capability advertisement. That enables node N1 to know which segment nodes are capable of responding to the ICMPv6 echo request. Node N1 processes the echo responses and presents data to the user, accordingly.

Please note that segment-by-segment ping can be used to address proof of transit use-case.

4.2. Error Reporting

Any IPv6 node can use ICMPv6 control messages to report packet processing errors to the host that originated the datagram packet. To name a few such scenarios:

- If the router receives an undeliverable IP datagram, or
- If the router receives a packet with a Hop Limit of zero, or
- If the router receives a packet such that if the router decrements the packet's Hop Limit it becomes zero, or
- If the router receives a packet with problem with a field in the IPv6 header or the extension headers such that it cannot complete processing the packet, or
- If the router cannot forward a packet because the packet is larger than the MTU of the outgoing link.

In the scenarios listed above, the ICMPv6 response also contains the IP header, IP extension headers and leading payload octets of the "original datagram" to which the ICMPv6 message is a response. Specifically, the "Destination Unreachable Message", "Time Exceeded Message", "Packet Too Big Message" and "Parameter Problem Message" ICMPV6 messages can contain as much of the invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU [RFC4443], [RFC4884]. In an SRv6 network, the copy of the invoking packet contains the SR header. The packet originator can use this information for diagnostic purposes. For example, traceroute can use this information as detailed in the following.

4.3. Traceroute

There is no hardware or software change required for traceroute operation at the classic IPv6 nodes in an SRv6 network. That includes the classic IPv6 node with ingress, egress or transit roles. Furthermore, no protocol changes are required to the standard traceroute operations. In other words, existing traceroute mechanisms work seamlessly in the SRv6 networks.

The following subsections outline some use cases of the traceroute in the SRv6 networks.

4.3.1. Classic Traceroute

The existing mechanism to traceroute a remote IP prefix, along the shortest path, continues to work without any modification. The initiator may be an SRv6 node or a classic IPv6 node. Similarly, the egress or transit may be an SRv6 node or a classic IPv6 node.

If an SRv6 capable ingress node wants to traceroute to IPv6 prefix via an arbitrary segment list <S1, S2, S3>, it needs to initiate traceroute probe with an SR header containing the SID list <S1, S2, S3>. That is illustrated using the topology in Figure 1. Assume all the links have IGP metric 10 except both links between node2 and node3, which have IGP metric set to 100. User issues a traceroute from node N1 to a loopback of node 5, via segment list <B:2:C31, B:4:C52>. Figure 3 contains sample output for the traceroute request.

```
> traceroute A:5:: via segment-list B:2:C31, B:4:C52

Tracing the route to B5::

 1  2001:DB8:1:2:21:: 0.512 msec 0.425 msec 0.374 msec
    SRH: (A:5::, B:4:C52, B:2:C31, SL=2)

 2  2001:DB8:2:3:31:: 0.721 msec 0.810 msec 0.795 msec
    SRH: (A:5::, B:4:C52, B:2:C31, SL=1)

 3  2001:DB8:3:4::41:: 0.921 msec 0.816 msec 0.759 msec
    SRH: (A:5::, B:4:C52, B:2:C31, SL=1)

 4  2001:DB8:4:5::52:: 0.879 msec 0.916 msec 1.024 msec
```

Figure 3 A sample traceroute output at an SRv6 capable node

Please note that information for hop2 is returned by N3, which is a classic IPv6 node. Nonetheless, the ingress node is able to display SR header contents as the packet travels through the IPv6 classic node. This is because the "Time Exceeded Message" ICMPv6 message can contain as much of the invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU [RFC4443]. The SR header is also included in these ICMPv6 messages initiated by the classic IPv6 transit nodes that are not running SRv6 software. Specifically, a node generating ICMPv6 message containing a copy of the invoking packet does not need to understand the extension header(s) in the invoking packet.

The segment list information returned for hop1 is returned by N2, which is an SRv6 capable node. Just like for hop2, the ingress node is able to display SR header contents for hop1.

There is no difference in processing of the traceroute probe at an IPv6 classic node and an SRv6 capable node. Similarly, both IPv6 classic and SRv6 capable nodes use the address of the interface on which probe was received as the source address in the ICMPv6

response. ICMP extensions defined in [RFC5837] can be used to also display information about the IP interface through which the datagram would have been forwarded had it been forwardable, and the IP next hop to which the datagram would have been forwarded, the IP interface upon which a datagram arrived, the sub-IP component of an IP interface upon which a datagram arrived.

The information about the IP address of the incoming interface on which the traceroute probe was received by the reporting node is very useful. This information can also be used to verify if SID functions B:2:C31 and B:4:C52 are executed correctly by N2 and N4, respectively. Specifically, the information displayed for hop2 contains the incoming interface address 2001:DB8:2:3:31:: at N3. This matches with the expected interface bound to END.X function B:2:C31 (link3). Similarly, the information displayed for hop5 contains the incoming interface address 2001:DB8:4:5::52:: at N5. This matches with the expected interface bound to the END.X function B:4:C52 (link10).

4.3.2. Traceroute to a SID Function

The classic traceroute described in the previous section cannot be used to traceroute a remote SID function, as explained using an example in the following.

Consider the case where the user wants to traceroute the remote SID function B:4:C52, via B:2:C31, from node N1. The trace route fails at N4. This is because the node N4 trace route probe where next header is UDP or ICMPv6, instead of SRH (even though the hop limit is set to 1). To solve this problem, the
initiator needs to mark the ICMPv6 echo request as an OAM packet.

The OAM packets are identified either by setting the O-flag in SRH or by inserting the END.OTP SID at an appropriate place in the SRH.

In an SRv6 network, the user can exercise two flavors of the traceroute: hop-by-hop traceroute or overlay traceroute.

- In hop-by-hop traceroute, user gets responses from all nodes including classic IPv6 transit nodes, SRv6 capable transit nodes as well as SRv6 capable segment endpoints. E.g., consider the example where the user wants to traceroute to a remote SID function B:4:C52 , via B:2:C31, from node N1. The traceroute

output will also display information about node3, which is a transit (underlay) node.
- The overlay traceroute, on the other hand, does not trace the underlay nodes. In other words, the overlay traceroute only displays the nodes that acts as SRv6 segments along the route. I.e., in the example where the user wants to traceroute to a remote SID function B:4:C52 , via B:2:C31, from node N1, the overlay traceroute would only display the traceroute information from node N2 and node N2 and will not display information from node 3.

4.3.2.1. Hop-by-hop traceroute using END.OP/ END.OTP

In this section, hop-by-hop traceroute to a SID function is exemplified using UDP probes. However, the procedure is equally applicable to other implementation of traceroute mechanism. Furthermore, the illustration uses the END.OTP SID but the procedures are equally applicable to the END.OP SID

Consider the same example where the user wants to traceroute to a remote SID function B:4:C52 , via B:2:C31, from node N1. To force a punt of the traceroute probe only at the node N4, node N1 inserts the END.OTP SID just before the target SID B:4:C52 in the SRH. The traceroute probe is processed at the individual nodes along the path as follows.

- Node N1 initiates a traceroute probe packet with a monotonically increasing value of hop count and SRH as follows (A:1::, B:2:C31)(B:4:C52, B:4:OTP, B:2:C31; SL=2; NH=UDP)(Traceroute probe).
- When node N2 receives the packet with hop-count = 1, it processes the hop count expiry. Specifically, the node N2 responses with the ICMPv6 message (Type: "Time Exceeded", Code: "Time to Live exceeded in Transit").
- When Node N2 receives the packet with hop-count > 1, it performs the standard SRH processing. Specifically, it executes the END.X function (B:2:C31) on the traceroute probe.
- When node N3, which is a classic IPv6 node, receives the packet (A:1::, B:4:OTP)(B:4:C52, B:4:OTP, B:2:C31 ; HC=1, SL=1; NH=UDP)(Traceroute probe) with hop-count = 1, it processes the hop count expiry. Specifically, the node N3 responses with the ICMPv6 message (Type: "Time Exceeded", Code: "Time to Live exceeded in Transit").
- When node N3, which is a classic IPv6 node, receives the packet with hop-count > 1, it performs the standard IPv6 processing. Specifically, it forwards the traceroute probe based on DA B:4:OTP in the IPv6 header.

- When node N4 receives the packet (A:1::, B:4:OTP)(B:4:C52, B:4:OTP, B:2:C31 ; SL=1; HC=1, NH=UDP)(Traceroute probe), it processes the END.OTP SID, as described in the pseudocode in Section 3. The packet gets punted to the traceroute process for processing. The traceroute process checks if the next SID in SRH (the target SID B:4:C52) is locally programmed. If the target SID B:4:C52 is locally programmed, node N4 responses with the ICMPv6 message (Type: Destination unreachable, Code: Port Unreachable). If the target SID B:4:C52 is not a local SID, node N4 silently drops the traceroute probe.

Figure 4 displays a sample traceroute output for this example.

```
> traceroute srv6 B:4:C52 via segment-list B:2:C31

Tracing the route to SID function B:4:C52

 1  2001:DB8:1:2:21 0.512 msec 0.425 msec 0.374 msec
    SRH: (B:4:C52, B:4:OTP, B:2:C31; SL=2)

 2  2001:DB8:2:3:31 0.721 msec 0.810 msec 0.795 msec
    SRH: (B:4:C52, B:4:OTP, B:2:C31; SL=1)

 3  2001:DB8:3:4::41 0.921 msec 0.816 msec 0.759 msec
    SRH: (B:4:C52, B:4:OTP, B:2:C31; SL=1)
```

Figure 4 A sample output for hop-by-hop traceroute to a SID function

4.3.2.2. Tracing SRv6 Overlay

The overlay traceroute does not trace the underlay nodes, i.e., only displays the nodes that acts as SRv6 segments along the path. This is achieved by setting the SRH.Flags.O bit.

In this section, overlay traceroute to a SID function is exemplified using UDP probes. However, the procedure is equally applicable to other implementation of traceroute mechanism.

Consider the same example where the user wants to traceroute to a remote SID function B:4:C52 , via B:2:C31, from node N1.

- Node N1 initiates a traceroute probe with SRH as follows (A:1::, B:2:C31)(B:4:C52, B:2:C31; HC=64, SL=1, Flags.O=1; NH=UDP)(Traceroute Probe). Please note that the hop-count is

set to 64 to skip the underlay nodes from tracing. The O-flag in SRH is set to make the overlay nodes (nodes processing the SRH) respond.

- When node N2 receives the packet (A:1::, B:2:C31)(B:4:C52, B:2:C31; SL=1, HC=64, Flags.O=1; NH=UDP)(Traceroute Probe), it processes the O-flag in SRH, as described in the pseudocode in Section 3. A time-stamped copy of the packet gets punted to the traceroute process for processing. Node N2 continues to apply the B:2:C31 SID function on the original packet and forwards it, accordingly. As SRH.Flags.O=1, Node N2 also disables the PSP flavor, i.e., does not remove the SRH. The traceroute process at node N2 checks if its local SID (B:2:C31) is locally programmed. If the SID is not locally programmed, it silently drops the packet. Otherwise, it performs the egress check by looking at the SL value in SRH.
- As SL is not equal to zero (i.e., it's not egress node), node N2 responses with the ICMPv6 message (Type: "SRv6 OAM (TBA)", Code: "O-flag punt at Transit (TBA)"). Please note that, as mentioned in Section 3, if node N2 does not support the O-flag, it simply ignores it and processes the local SID, B:2:C31.
- When node N3 receives the packet (A:1::, B:4:C52)(B:4:C52, B:2:C31; SL=0, HC=63, Flags.O=1; NH=UDP)(Traceroute Probe), performs the standard IPv6 processing. Specifically, it forwards the traceroute probe based on DA B:4:C52 in the IPv6 header. Please note that there is no hop-count expiration at the transit nodes.
- When node N4 receives the packet (A:1::, B:4:C52)(B:4:C52, B:2:C31; SL=0, HC=62, Flags.O=1; NH=UDP)(Traceroute Probe), it processes the O-flag in SRH, as described in the pseudocode in Section 3. A time-stamped copy of the packet gets punted to the traceroute process for processing. The traceroute process at node N4 checks if its local SID (B:2:C31) is locally programmed. If the SID is not locally programmed, it silently drops the packet. Otherwise, it performs the egress check by looking at the SL value in SRH. As SL is equal to zero (i.e., N4 is the egress node), node N4 tries to consume the UDP probe. As UDP probe is set to access an invalid port, the node N4 responses with the ICMPv6 message (Type: Destination unreachable, Code: Port Unreachable).

Figure 5 displays a sample overlay traceroute output for this example. Please note that the underlay node N3 does not appear in the output.

```
> traceroute srv6 B:4:C52 via segment-list B:2:C31

Tracing the route to SID function B:4:C52
```

```
1  2001:DB8:1:2:21:: 0.512 msec 0.425 msec 0.374 msec
   SRH: (B:4:C52, B:4:OTP, B:2:C31; SL=2)

2  2001:DB8:3:4::41:: 0.921 msec 0.816 msec 0.759 msec
   SRH: (B:4:C52, B:4:OTP, B:2:C31; SL=1)
```

Figure 5 A sample output for overlay traceroute to a SID function

4.5. Monitoring of SRv6 Paths

In the recent past, network operators are interested in performing network OAM functions in a centralized manner. Various data models like YANG are available to collect data from the network and manage it from a centralized entity.

SR technology enables a centralized OAM entity to perform path monitoring from centralized OAM entity without control plane intervention on monitored nodes. [I.D-draft-ietf-spring-oam-usecase] describes such a centralized OAM mechanism. Specifically, the draft describes a procedure that can be used to perform path continuity check between any nodes within an SR domain from a centralized monitoring system, with minimal or no control plane intervene on the nodes. However, the draft focuses on SR networks with MPLS data plane. The same concept applies to the SRv6 networks. This document describes how the concept can be used to perform path monitoring in an SRv6 network. This document describes how the concept can be used to perform path monitoring in an SRv6 network as follows.

In the above reference topology, N100 is the centralized monitoring system implementing an END function B:100:1::. In order to verify a segment list <B:2:C31, B:4:C52>, N100 generates a probe packet with SRH set to (B:100:1::, B:4:C52, B:2:C31, SL=2). The controller routes the probe packet towards the first segment, which is B:2:C31. N2 performs the standard SRH processing and forward it over link3 with the DA of IPv6 packet set to B:4:C52. N4 also performs the normal SRH processing and forward it over link10 with the DA of IPv6 packet set to B:100:1::. This makes the probe loops back to the centralized monitoring system.

In the reference topology in Figure 1, N100 uses an IGP protocol like OSPF or ISIS to get the topology view within the IGP domain. N100 can also use BGP-LS to get the complete view of an inter-domain topology. In other words, the controller leverages the visibility of the topology to monitor the paths between the various endpoints without control plane intervention required at the monitored nodes.

5. Security Considerations

This document does not define any new protocol extensions and relies on existing procedures defined for ICMP. This document does not impose any additional security challenges to be considered beyond security considerations described in [RFC4884], [RFC4443], [RFC792] and RFCs that updates these RFCs.

6. IANA Considerations

6.1. ICMPv6 type Numbers Registry

This document defines one ICMPv6 Message, a type that has been allocated from the "ICMPv6 'type' Numbers" registry of [RFC4443].

Specifically, it requests to add the following to the "ICMPv6 Type Numbers" registry:

TBA (suggested value: 162) SRv6 OAM Message.

The document also requests the creation of a new IANA registry to the

"ICMPv6 'Code' Fields" against the "ICMPv6 Type Numbers TBA - SRv6 OAM Message" with the following codes:

Code	Name	Reference
0	No Error	This document
1	SID is not locally implemented	This document
2	O-flag punt at Transit	This document

6.3. SRv6 OAM Endpoint Types

This I-D requests to IANA to allocate, within the "SRv6 Endpoint Behaviors Registry" sub-registry belonging to the top-level "Segment-routing with IPv6 dataplane (SRv6) Parameters" registry [I-D.filsfils-spring-srv6-network-programming], the following allocations:

Value (Suggested Value)	Endpoint Behavior	Reference
TBA (30)	End.OP	[This.ID]
TBA (31)	End.OTP	[This.ID]

7. References

7.1. Normative References

[RFC792] J. Postel, "Internet Control Message Protocol", RFC 792, September 1981.

[RFC4443] A. Conta, S. Deering, M. Gupta, Ed., "Internet Control

Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4884] R. Bonica, D. Gan, D. Tappan, C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, April 2007.

[RFC5837] A. Atlas, Ed., R. Bonica, Ed., C. Pignataro, Ed., N. Shen, JR. Rivers, "Extending ICMP for Interface and Next-Hop Identification", RFC 5837, April 2010.

[I-D.filsfils-spring-srv6-network-programming] C. Filsfils, et al., "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming, work in progress.

[I-D.6man-segment-routing-header] Previdi, S., Filsfils, et al, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header, work in progress.

7.2. Informative References

[I-D.bashandy-isis-srv6-extensions] IS-IS Extensions to Support Routing over IPv6 Dataplane. L. Ginsberg, P. Psenak, C. Filsfils, A. Bashandy, B. Decraene, Z. Hu, draft-bashandy-isis-srv6-extensions, work in progress.

[I-D.dawra-idr-bgpls-srv6-ext] G. Dawra, C. Filsfils, K. Talaulikar, et al., BGP Link State extensions for IPv6 Segment Routing (SRv6), draft-dawra-idr-bgpls-srv6-ext, work in progress.

[I-D.ietf-spring-oam-usecase] A Scalable and Topology-Aware MPLS Dataplane Monitoring System. R. Geib, C. Filsfils, C. Pignataro, N. Kumar, draft-ietf-spring-oam-usecase, work in progress.

[I-D.brockners-inband-oam-data] F. Brockners, et al., "Data Formats for In-situ OAM", draft-brockners-inband-oam-data, work in progress.

[I-D.brockners-inband-oam-transport] F.Brockners, at al., "Encapsulations for In-situ OAM Data", draft-brockners-inband-oam-transport, work in progress.

[I-D.brockners-inband-oam-requirements] F.Brockners, et al., "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements, work in progress.

[I-D.spring-segment-routing-policy] Filsfils, C., et al., "Segment Routing Policy for Traffic Engineering", draft-filsfils-spring-segment-routing-policy, work in progress.

8. Acknowledgments

To be added.

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Email: cfilsfil@cisco.com

Zafar Ali
Cisco Systems, Inc.
Email: zali@cisco.com

Nagendra Kumar
Cisco Systems, Inc.
Email: naikumar@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
Email: cpignata@cisco.com

Faisal Iqbal
Cisco Systems, Inc.
Email: faiqbal@cisco.com

Rakesh Gandhi
Cisco Systems, Inc.
Canada
Email: rgandhi@cisco.com

John Leddy
Comcast
Email: John_Leddy@cable.comcast.com

Robert Raszuk
Bloomberg LP
731 Lexington Ave
New York City, NY10022, USA
Email: robert@raszuk.net

Satoru Matsushima
SoftBank
Japan
Email: satoru.matsushima@g.softbank.co.jp

Daniel Voyer
Bell Canada
Email: daniel.voyer@bell.ca

Gaurav Dawra
LinkedIn
Email: gdawra.ietf@gmail.com

Bart Peirens
Proximus
Email: bart.peirens@proximus.com

Mach Chen
Huawei
Email: mach.chen@huawei.com

Gaurav Naik
Drexel University
United States of America
Email: gn@drexel.edu

INTERNET-DRAFT
Intended Status: Standard
Expires: July 2018

T. Herbert
Quantonium

January 15, 2018

ICMPv6 errors for discarding packets due to processing limits
draft-herbert-6man-icmp-limits-03

Abstract

Network nodes may discard packets if they are unable to process protocol headers of packets due to processing constraints or limits. When such packets are dropped, the sender receives no indication so it cannot take action to address the cause of discarded packets. This document defines ICMPv6 errors that can be sent by a node that discards packets because it is unable to process the protocol headers. A node that receives such an ICMPv6 error may be able to modify what it sends in future packets to avoid subsequent packet discards.

Table of Contents

1 Introduction

This document specifies ICMPv6 errors that can be sent when a node discards a packet due to it being unable to process the necessary protocol headers because of processing constraints or limits. New ICMPv6 code points are defined as an update to [RFC4443].

Four of the errors are specific to processing limits of extension headers; another error is used when the aggregate protocol headers in a packet exceed the processing limits of a node.

1.1 Extension header limits

With IPv6, optional internet-layer information is carried in one or more IPv6 Extension Headers [RFC8200]. Extension Headers are placed between the IPv6 header and the Upper-Layer Header in a packet. The term "Header Chain" refers collectively to the IPv6 header, Extension Headers, and Upper-Layer Headers occurring in a packet. Individual extension headers may have a length of 2048 and must fit into one MTU. Destination Options and Hop by Hop Options contain a list options in Type-length-value (TLV) format. Each option includes a length of the data field in octets, and the minimum size of an option (non-pad type) is two bytes and the maximum length is 257 bytes. The number of options in an extension header is only limited by the length of the extension header and MTU. Options may also be skipped over by a receiver if they are unknown and the Option Type indicates to skip (first two bits are 00).

Per [RFC8200], except for Hop by Hop options, extension headers are not examined or processed by intermediate nodes. Many intermediate nodes, however, do examine extension header for various purposes. For instance, a node may examine all extension headers to locate the transport header of packet in order to implement transport layer filtering or to track connections to implement a stateful firewall.

Destination hosts are expected to process all extensions headers and options in Hop-by-Hop and Destination Options.

Due to the variable lengths, high limits of lengths of extension headers, or potential for Denial of Service attack; many devices impose operational limits of extension headers in packets they can process. [RFC7045] discusses the requirements of intermediate nodes that discard packets because of unrecognized extension headers. When a limit is exceeded, the typical behavior is to silently discard a packet. The limits are non-standard and may be configurable per implementation. Both intermediate nodes and end hosts may institute such limits on extension header processing.

This document defines three Parameter Problem codes and extends the applicably of an existing code that are sent by a node that discards a packet due to processing limits of extension headers being exceeded. A source host that receives an ICMPv6 error can modify the use of extension headers in subsequent packets to the destination in order to avoid further occurrences of packets being discarded.

1.2 Aggregate header limits

Many hardware devices implement a parsing buffer of a fixed sized to process packets. The parsing buffer is expected to contain all the headers (often up to a transport layer header for filtering) that a device needs to examine. Parsing buffers have been implemented with various sizes (512 bytes is common, some devices have smaller sizes).

When the aggregate length of headers in a packet exceeds the size of the parsing buffer, a device will typically either discard the packet or defer processing to a software slow path. In either case, no indication of a problem is sent back to the sender.

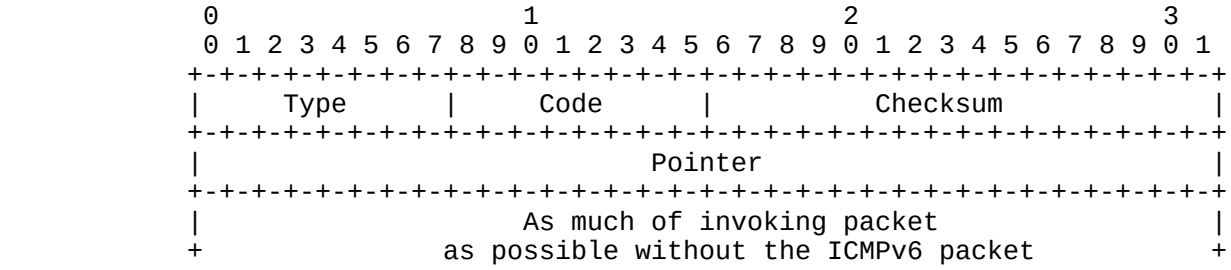
This document defines one code for ICMPv6 Destination Unreachable that is sent by a node that is unable to process the headers of a packet due to the aggregate size of the packet headers exceeding a processing limit (e.g. exceeding the size of a parsing buffer). A source host that receives an ICMPv6 error can modify the headers used in subsequent packets to try to avoid further occurrences of packets being discarded or relegated to a slow path.

2 ICMPv6 errors for extension header limits

Three new codes are defined for Parameter Problem type and applicability of one existing code is extended for ICMPv6 errors for extension header limits.

2.1 Format

The format of the ICMPv6 message for an extension header limit exceeded error is:



| exceeding the minimum IPv6 MTU [IPv6] |

IPv6 Fields:

Destination Address
Copied from the Source Address field of the invoking packet.

ICMPv6 Fields:

Type
4 (Parameter Problem type)

Code (pertinent to this specification)
1 - Unrecognized Next Header type encountered
4 - Extension header too big
5 - Extension header chain too long
6 - Too many options in extension header

Pointer
Identifies the octet offset within the invoking packet where the problem occurred.

The pointer will point beyond the end of the ICMPv6 packet if the field having a problem is beyond what can fit in the maximum size of an ICMPv6 error message.

2.2 Unrecognized Next Header type encountered (code 1)

[RFC8200] specifies that a destination host should send an "unrecognized next header type" when a Next Header value is unrecognized in a packet. This document extends this to allow intermediate nodes to send this same error for a packet that is discarded because a node does not recognize a Next Header type.

This code SHOULD be sent by an intermediate node that discards a packet because it encounters a Next Header type that is unknown in its examination. The ICMPv6 Pointer field is set to the offset of the unrecognized value within the original packet.

Note that when the original sender receives the ICMPv6 error it can differentiate between the message being sent by a destination host, per [RFC4443], and an error sent by an intermediate host based on matching the source address of the ICMPv6 packet and the destination address of the packet in the ICMPv6 data.

2.3 Extension header too big (code 4)

An ICMPv6 Parameter Problem with code for "extension header too big"

SHOULD be sent when a node discards a packet because the size of an extension header exceeds its processing limit. The ICMPv6 Pointer field is set to the offset of the first octet in the extension header that exceeds the limit.

2.4 Extension header chain too long (code 5)

An ICMPv6 Parameter Problem with code for "extension header chain too long" SHOULD be sent when a node discards a packet with an extension header chain because an extension header chains exceeds it processing limit.

There are two different limits that might be applied: a limit on the total size in octets of the header chain, and a limit on the number of extension headers in the chain. This error code is used in both cases. In the case that the a size limit is exceeded, the ICMPv6 Pointer is set to first octet beyond the limit. In the case that the number of extension headers is exceeded, the ICMPv6 Pointer is set to the offset of first octet of the first extension header that is beyond the limit.

2.5 Too many options in extension header (code 6)

An ICMPv6 Parameter Problem with code for "too many options in extension header" SHOULD be sent when a node discards a packet with an extension header that has a number of options that exceed the processing limits of the node. This code is applicable for Destination options or Hop-by-Hop options. The ICMPv6 Pointer field is set to the first octet of the first option that exceeds the limit.

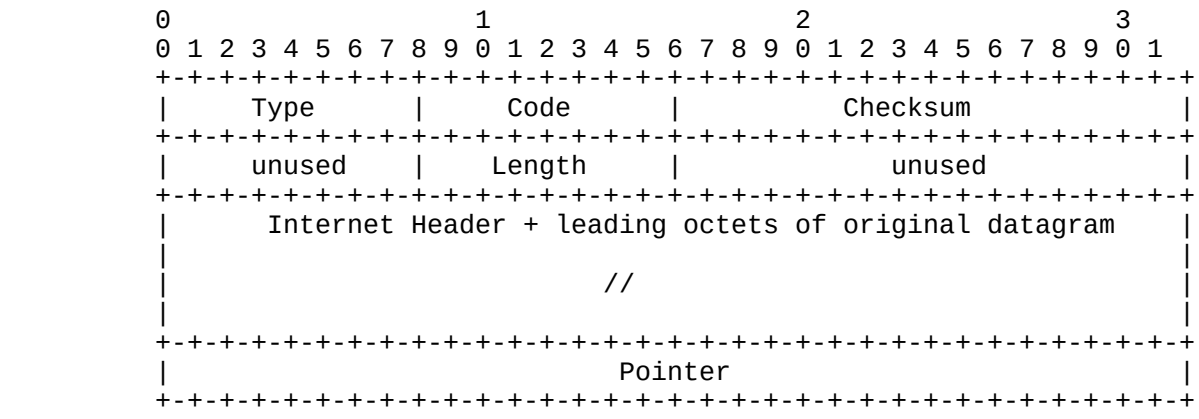
3 ICMPv6 error for aggregate header limits

One code is defined for Destination Unreach type for aggregate header limits.

3.1 Format

The error for aggregate header limits employs a multi-part ICMPv6 message format as defined in [RFC4884]. The extended structure contains a pointer to the octet beyond the limit.

The format of the ICMPv6 message for an aggregate header limit exceeded is:



IPv6 Fields:

Destination Address
Copied from the Source Address field of the invoking packet.

ICMPv6 Fields:

Type
1 (Destination Unreachable type)

Code (pertinent to this specification)
8 - Headers too long

Length
Length of the "original datagram" measured in 64 bit words

Pointer
Identifies the octet offset within the invoking packet where a limit was exceeded.

The pointer will point beyond the end of the original datagram if the field exceeding the limit is beyond what can fit in the maximum size of an ICMPv6 error message.

3.2 Usage

An ICMPv6 Destination Unreachable error with code for "headers too long" SHOULD be sent when a node discards a packet because the aggregate length of headers in the packet exceeds the processing limits of the node. The Pointer in the extended ICMPv6 structure is set to the offset of the first octet that exceeds the limit.

4 Operation

Nodes that send or receive ICMPv6 errors due to header processing limits MUST generally comply with ICMPv6 processing as specified in [RFC4443].

4.1 Priority of reporting

More than one ICMPv6 error may be applicable to report for a packet. For instance, the number of extension headers in a packet might exceed a limit, and the aggregate length of protocol headers might also exceed a limit. Only one ICMPv6 error should be sent for a packet, so a priority is defined to determine which error to report.

The reporting priority of ICMPv6 errors for processing limits is from highest to lowest priority:

1) Real error (existing codes)

2) Unrecognized Next Header type encountered by an intermediate node

3) Too many options in an extension header

4) Extension header too big

5) Extension header chain too long for number of extension headers limit exceeded

6) Extension header chain too long for size of the extension header chain exceeding a limit

7) Headers too long

4.2 Host response

When a source host receives an ICMPv6 error for a processing limit being exceeded, it SHOULD verify the ICMPv6 error is valid and take an appropriate action.

The ICMPv6 error SHOULD be logged with sufficient detail for debugging packet loss. The details of the error, including the addresses and the offending extension header or data, should be retained. This would be useful for instance to debug when a node is mis-configured and unexpectedly discarding packets, or when a new extension header is being deployed.

A host MAY modify its usage of protocol headers in subsequent packets

to avoid repeated occurrences of the same error.

For ICMPv6 errors cause by extension header limits being exceeded:

* An error SHOULD be reported to an application if the application enabled extension headers for its traffic. The application MAY either terminate a connection if extension headers are required, stop using extension headers in packets to the destination indicated in packet of the ICMPv6 error, or attempt modify its use of extension headers or headers to avoid the packet drop.

* A host system SHOULD take action if it is automatically inserting extension headers into packets unbeknownst to the application. The host system SHOULD either stop using extension headers or modify its used of extension headers for subsequent packets sent to the destination indicated in the packet of the ICMPv6 error.

5 Security Considerations

This document does not introduce any new security concerns for use of ICMPv6 errors. The security considerations for ICMPv6 described in [RFC4443] are applicable.

6 IANA Considerations

6.1 Parameter Problem codes

IANA is requested to assign the following codes for ICMPv6 type 4 "Parameter Problem":

4 - Extension header too big

5 - Extension header chain too long

6 - Too many options in extension header

6.2 Destination Unreachable codes

IANA is requested to assign the following codes for ICMPv6 type 1 "Destination Unreachable":

8 - Headers too long

8 Acknowledgments

The authors would like to thank Ron Bonica, Bob Hinden for their comments and suggestions that improved this document.

7 References

7.1 Normative References

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <http://www.rfc-editor.org/info/rfc7045>.

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <https://www.rfc-editor.org/info/rfc4884>.

7.2 Informative References

Author's Address

Tom Herbert
Quantonium
Santa Clara, CA
USA

Email: tom@herbertland.com

INTERNET-DRAFT
Intended Status: Standard
Expires: March 10, 2019

Tom Herbert
Quantonium

September 14, 2018

Updates to Requirements for IPv6 Options
draft-herbert-ipv6-update-opts-00

Abstract

This document updates requirements for IPv6 Destination and Hop-by-Hop Options. The requirements that option type and option length cannot change en route, as well as the requirements that options cannot be added or removed, are made explicit. The meaning and requirements of a Destination Option marked as changeable are clarified. Finally, the requirement that all destinations listed in a Routing header must process options in a Destination Options header preceding the Routing header is relaxed.

Table of Contents

1 Introduction

[RFC8200] defines Hop-by-Hop and Destination Options. This document clarifies requirements for changing, adding, or removing options in a packet en route to its final destination. It also relaxes the requirement that Destination Options preceding a Routing header must be processed by all destinations listed in the Routing header.

[RFC8200] specifies that "The third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en route to the packet's final destination." It is implicit in this requirement that neither the Option Type nor Option Data Length can change en route to the packet's destination. It also follows that options cannot be added or removed while a packet is en route. This document makes these requirements explicit.

Per [RFC8200], Destination Options may be marked as changeable (the third-highest-order bit of the Option Type for the Destination Option is set). [RFC8200] also states that with the exception of Hop-by-Hop options, extension headers are not processed except by the destination node. It follows that the only possible case that a Destination Option may be modified en route is by a node that is one of destinations to be visited in a Routing header. This document clarifies this requirement.

Per [RFC8200], if a Destination Options header precedes a Routing header, then all of the destinations listed in the Routing header must process the Destination Options. This document proposes to relax that requirement by allowing nodes listed in the Routing header to ignore Destination Options that precede the Routing header. The motivation for this is similar to that of relaxing the requirement that all intermediate nodes process Hop-by-Hop options in [RFC8200]. Intermediate destination nodes may be closer in taxonomy to switches and routers than end hosts, so it follows that they may have similar processing constraints in efficiently processing extension headers and TLVs. Those constraints could lead to similar ad hoc behaviors for processing packets with options-- some implementations have dropped packets with options, others have relegated them to slow path processing. In any case, such behaviors at even a few nodes can essentially render options unusable. Allowing nodes to ignore options retains the primary value and usability of Destination Options preceding a Routing header. Nodes that are not interested in them can ignore them, nodes that fully support them can process them.

2 Requirements for adding, removing, or changing options

This section clarifies requirements of [RFC8200] for changing, adding, or removing Destination Options or Hop-by-Hop Options.

The Option Type of an option MUST NOT be changed en route to a packet's final destination. Note that this precludes changing the high order bits of an Option Type which indicate a changeable option or the action to take for an unknown option.

The Option Data Length of an option MUST NOT be changed en route to a packet's final destination. If the third-highest-order bit of the Option Type is set indicating that the Option Data can change en route, then any changes MUST be to the existing Option Data and the Option Length MUST be preserved. Note, if the Option Data Length is zero then the option cannot be modified in any way.

Options MUST NOT be added to or removed from a packet en route to its final destination. This requirement precludes adding or removing options within an existing extension header, as well as adding or removing a Destination or Hop-by-Hop extension headers in a packet.

Note that in the case that a routing header is present, the "final destination" refers to the final destination listed to visit in the routing header. At intermediate destinations of a routing header, the packet is considered en route to the final destination, so that requirements about changing a packet en route to its final destination are applicable.

3 Requirements for changeable Destination Options

If a Destination Option in a Destination Options header that precedes a Routing header is marked as changeable (the third-highest order bit of the option type is set), then the Option Data may be changed by any destination node en route to the final destination. Specifically, the node for the initial destination address as well as any nodes to visit as listed in the Routing header may change the Option Data.

If a Destination Option is marked as changeable (the third-highest order bit of the option type is set) and is in a Destination Options header that follows a Routing header, or there is no Routing header present, then the Option Data cannot be changed en route. There are no nodes in the path that are permitted to change the Option Data. Note that the requirement when an Authentication header is present the entire Option Data field must be treated as zero-valued octets when computing or verifying the packet's authenticating value is still applicable.

4 Requirements for processing Destination Options

This section clarifies requirements of processing Destination Options with respect to its relationship to a Routing header.

Options in a Destination Options header that follow a Routing header, or are in a packet having no Routing header, MUST be processed by the destination node. In the case that a Routing header is present, the Destination Options that follow the Routing header MUST be processed by the final destination listed in the Routing header.

Options in a Destination Options header that precede a Routing header MAY be examined or processed by the original destination node and nodes listed to visit in the Routing header (including the final destination of the Routing Header). If a node does not process the options in a Destination Option header, then it MUST skip over the Destination Options header and continue to process the next header which is likely the Routing header.

5 Detecting that Destination Options precede a Routing header

As specified in requirements of this document, an implementation might process Destination Options differently depending on whether they precede a Routing header. Procedures are therefore needed to detect if Destination Options precede a Routing header.

An implementation MAY determine that Destination Options precede a Routing Header by inspecting the Next Header field of the Destination Option. If the Next Header field indicates a Routing Header, then the implementation can conclude that Destination Options precede a Routing Header. Note that this employs a heuristic based on the recommended ordering of extension headers of [RFC8200] in which the Routing header should immediately follow Destination Options before a Routing header.

An implementation MAY scan the packet to determine if a Routing header is present that follows a Destination Options header. If such a scan is performed, an implementation MUST NOT process any scanned extension headers beyond inspecting their Next Header and Header Ext Length fields. This requirement is necessary ensure that extension headers are strictly processed order as manadated by [RFC8200].

If a node is not able to determine that Destination Options precede a Routing header, the Destinations Options MUST be processed as though they do not precede a Routing header. In this case, a destination node, regardless whether it is an intermediate or final destination, MUST process the Destination Options and MUST NOT change any Destination Options even if they are marked as changeable.

6 Security Considerations

Relaxing the requirement that Destination Options preceding a Routing header can be ignored by intermediate destination nodes should not pose any new security risk. It should be noted that any security mechanism specified in a Destination Option should take into account that not all intermediate destinations would necessarily process the security option.

7 IANA Considerations

There are no IANA considerations in this specification.

8 References

8.1 Normative References

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

Author's Address

Tom Herbert
Quantonium
Santa Clara, CA
USA

Email: tom@quantonium.net

INTERNET-DRAFT
Intended Status: Standard
Expires: September 2, 2019

Tom Herbert
Quantonium

March 1, 2019

Updates to Requirements for IPv6 Options
draft-herbert-ipv6-update-opts-01

Abstract

This document updates requirements for IPv6 Destination and Hop-by-Hop Options. The requirements that option type and option length cannot change en route, as well as the requirements that options cannot be added or removed, are made explicit. The meaning and requirements of a Destination Option marked as changeable are clarified. Finally, the requirement that all destinations listed in a Routing header must process options in a Destination Options header preceding the Routing header is relaxed.

Table of Contents

1 Introduction

[RFC8200] specifies Hop-by-Hop and Destination Options. This document clarifies requirements for changing, adding, or removing options in a packet en route to its final destination. It also relaxes the requirement that Destination Options preceding a Routing header must be processed by all destinations listed in the Routing header.

[RFC8200] states that "The third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en route to the packet's final destination." It is implicit in this requirement that neither the Option Type nor Option Data Length can change en route to the packet's destination. It also follows that options cannot be added or removed while a packet is en route. This document makes these requirements explicit.

Per [RFC8200], a Destination Option is marked as changeable en route when the third-highest-order bit of the Option Type for the Destination Option is set. [RFC8200] also states that with the exception of Hop-by-Hop options, extension headers are not examined or processed by any intermediate nodes in the path. It follows that the only possible case that a Destination Option may be modified en route is by a node that is one of destinations to be visited in a Routing header. This document clarifies this requirement.

Per [RFC8200], if a Destination Options header precedes a Routing header, then all of the destinations listed in the Routing header must process the Destination Options. This document proposes to relax that requirement by allowing nodes listed in the Routing header to ignore Destination Options that precede the Routing header. The motivation for this is similar to that of relaxing the requirement that all intermediate nodes process Hop-by-Hop options in [RFC8200]. Intermediate destination nodes may be closer in taxonomy to switches and routers than end hosts, so it follows that they may have similar processing constraints in efficiently processing extension headers and TLVs. Those constraints could lead to similar ad hoc behaviors for processing packets with options-- some implementations have dropped packets with options, others have relegated them to slow path processing. In any case, such behaviors, even at a few nodes, can essentially render options unusable. Allowing nodes to ignore options retains the primary value and usability of Destination Options preceding a Routing header. Nodes that are not interested in them can ignore them, nodes that fully support them can process them.

2 Requirements for adding, removing, or changing options

This section clarifies requirements of [RFC8200] for adding, removing, or changing Destination Options or Hop-by-Hop Options.

The Option Type of an option MUST NOT be changed en route to a packet's final destination. Note that this precludes changing the high order bits of an Option Type which indicate a changeable option or the action to take for an unknown option.

The Option Data Length of an option MUST NOT be changed en route to a packet's final destination. If the third-highest-order bit of the Option Type is set indicating that the Option Data can change en route, then any changes MUST be to the existing Option Data and the Option Length MUST be preserved. Note, if the Option Data Length is zero then the option cannot be modified in any way.

Options MUST NOT be added to or removed from a packet en route to its final destination. This requirement precludes adding or removing options within an existing extension header, as well as adding or removing Destination or Hop-by-Hop extension headers in a packet.

Note that in the case that a routing header is present, the "final destination" refers to the final destination listed to visit in the routing header. At intermediate destinations of a routing header, the packet is considered en route to its final destination, so that requirements about changing Destination Options en route are applicable.

3 Requirements for changeable Destination Options

If a Destination Option in a Destination Options header that precedes a Routing header is marked as changeable, that is the third-highest order bit of the Option Type is set, then the Option Data may be changed by any intermediate destination node en route to the final destination. Specifically, the node for the initial destination address as well as any nodes to visit as listed in the Routing header may change the Option Data.

If a Destination Option is marked as changeable and is in a Destination Options header that follows a Routing header, or there is no Routing header present, then the Option Data cannot be changed en route. There are no nodes in the path that are permitted to change the Option Data. Note that the requirement when an Authentication header is present the entire Option Data field of a changeable option must be treated as zero-valued octets when computing or verifying the packet's authenticating value is still applicable.

4 Requirements for processing Destination Options

This section clarifies requirements of processing Destination Options with respect to its relationship to a Routing header.

Options in a Destination Options header that follow a Routing header, or are in a packet having no Routing header, MUST be processed by the destination node. In the case that a Routing header is present, the Destination Options that follow the Routing header MUST be processed by the final destination listed in the Routing header.

Options in a Destination Options header that precede a Routing header MAY be examined or processed by the original destination node and nodes listed to visit in the Routing header. If a node does not process the options in a Destination Option header, then it MUST skip over the Destination Options header and continue to process the next header which is likely the Routing header. The final destination of a packet, which is the last node listed in the Routing header, MUST process a Destination Options header that appears before the routing header. Any intermediate nodes that are not explicitly listed as intermediate destinations in the Routing header MUST NOT examine or process a Destination Options header preceding a Routing header.

5 Detecting that Destination Options precede a Routing header

As specified in requirements of this document, an implementation might process Destination Options differently depending on whether they precede a Routing header. Procedures are therefore needed to detect if a Destination Options header precedes a Routing header.

An implementation MAY determine that Destination Options precede a Routing header by inspecting the Next Header field of the Destination Options header. If the Next Header field indicates a Routing header, then the implementation can conclude that Destination Options precede a Routing Header. Note that this employs a heuristic based on the recommended ordering of extension headers of [RFC8200] in which the Routing header should immediately follow Destination Options header before a Routing header.

An implementation MAY scan the packet to determine if a Routing header is present that follows a Destination Options header. If such a scan is performed, an implementation MUST NOT process any scanned extension headers beyond inspecting their Next Header and Header Ext Length fields. This requirement is necessary ensure that extension headers are strictly processed order as manadated by [RFC8200].

If a node is unable to determine that Destination Options precede a Routing header, the Destinations Options MUST be processed as though

they do not precede a Routing header. In this case, a destination node, regardless whether it is an intermediate or final destination, MUST process the Destination Options and MUST NOT change any Destination Options even if they are marked as changeable.

6 Security Considerations

Relaxing the requirement that Destination Options preceding a Routing header can be ignored by intermediate destination nodes should not pose any new security risk. It should be noted that any security mechanism specified in a Destination Option should take into account that not all intermediate destinations would necessarily process the option in a Destination Options header that precedes a Routing header.

7 IANA Considerations

There are no IANA considerations in this specification.

8 References

8.1 Normative References

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

Author's Address

Tom Herbert
Quantonium
Santa Clara, CA
USA

Email: tom@quantonium.net

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 6, 2020

R. Hinden
Check Point Software
G. Fairhurst
University of Aberdeen
July 5, 2019

IPv6 Minimum Path MTU Hop-by-Hop Option
draft-hinden-6man-mtu-option-02

Abstract

This document specifies a new Hop-by-Hop IPv6 option that is used to record the minimum Path MTU along the forward path between a source to a destination host. This collects a minimum recorded MTU along the path to the destination. The value can then be communicated back to the source using the return Path MTU field in the option.

This Hop-by-Hop option is intended to be used in environments like Data Centers and on paths between Data Centers, to allow them to better take advantage of paths able to support a large Path MTU.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 6, 2020.

carefully, as they describe your rights and restrictions with respect to this document.

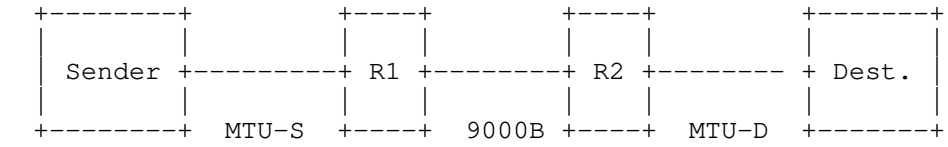
Table of Contents

1. Introduction

This draft proposes a new Hop-by-Hop Option to be used to record the minimum MTU along the forward path between the source and destination nodes. The source node creates a packet with this Hop-by-Hop Option and fills the Reported PMTU Field in the option with the value of the MTU for the outbound link that will be used to forward the packet towards the destination.

At each subsequent hop where the option is processed, the router compares the value of the Reported PMTU in the option and the MTU of its outgoing link. If the MTU of the outgoing link is less than the Reported PMTU specified in the option, it rewrites the value in the Option Data with the smaller value. When the packet arrives at the Destination node, the Destination node can send the minimum reported PMTU value back to the Source Node using the Return PMTU field in the option.

The figure below can be used to illustrate the operation of the method. In this case, the path between the Sender and Destination nodes comprises three links, the sender has a link MTU of size MTU-S, the link between routers R1 and R2 has an MTU of size 8 KBytes, and the final link to the destination has an MTU of size MTU-D.



The scenarios are described:

Scenario 1, considers all links to have an 9000 Byte MTU and the method is supported by both routers.

Scenario 2, considers the destination link to have an MTU of 1500 Byte. This is the smallest MTU, router R2 resets the reported PMTU to 1500 Byte and this is detected by the method. Had there been another smaller MTU at a link further along the path that supports the method, the lower PMTU would also have been detected.

Scenario 3, considers the case where the router preceding the smallest link does not support the method, and the method then fails to detect the actual PMTU. These scenarios are summarized in the table below. This scenario would also arise if the PTB message was not delivered to the sender.

	MTU-S	MTU-D	R1	R2	Rec PMTU	Note
1	9000B	9000B	H	H	9000 B	Endpoints attempt to use an 9000 B PMTU.
2	9000B	1500B	H	H	1500 B	Endpoints attempt to use a 1500 B PMTU.
3	9000B	1500B	H	-	9000 B	Endpoints attempt to use an 9000 B PMTU, but need to implement a method to fall back use a 1500 B PMTU.

IPv6 as specified in [RFC8200] allows nodes to optionally process Hop-by-Hop headers. Specifically from Section 4:

- The Hop-by-Hop Options header is not inserted or deleted, but may be examined or processed by any node along a packet's delivery path, until the packet reaches the node (or each of the set of

nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header. The Hop-by-Hop Options header, when present, must immediately follow the IPv6 header. Its presence is indicated by the value zero in the Next Header field of the IPv6 header.

- NOTE: While [RFC2460] required that all nodes must examine and process the Hop-by-Hop Options header, it is now expected that nodes along a packet's delivery path only examine and process the Hop-by-Hop Options header if explicitly configured to do so.

The Hop-by-Hop Option defined in this document is designed to take advantage of this property of how Hop-by-Hop options are processed. Nodes that do not support this Option SHOULD ignore them. This can mean that the value returned in the response message does not account for all links along a path.

2. Motivation and Problem Solved

The current state of Path MTU Discovery on the Internet is problematic. The problems with the mechanisms defined in [RFC8201] are known to not work well in all environments. Nodes in the middle of the network may not send ICMP Packet Too Big messages or they are rate limited to the point of not making them a useful mechanism.

This results in many connection defaulting to 1280 octets and makes it very difficult to take advantage of links with larger MTU where they exist. Applications that need to send large packets over UDP are forced to use IPv6 Fragmentation.

Transport encapsulations and network-layer tunnels reduce the PMTU available for a transport to use. For example, Network Virtualization Using Generic Routing Encapsulation (NVGRE) [RFC7637] encapsulates L2 packets in an outer IP header and does not allow IP Fragmentation.

The use of 10G Ethernet will not achieve it's potential because the packet per second rate will exceed what most nodes can send to achieve multi-gigabit rates if the packet size limited to 1280 octets. For example, the packet per second rate required to reach wire speed on a 10G Ethernet link with 1280 octet packets is about 977K packets per second (pps), vs. 139K pps for 9,000 octet packets. A significant difference.

The purpose of the this draft is to improve the situation by defining a mechanism that does not rely on nodes in the middle of the network to send ICMPv6 Packet Too Big messages, instead it provides the destination host information on the minimum Path MTU and it can send

this information back to the source host. This is expected to work better than the current RFC8201 based mechanisms.

3. Requirements Language

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

4. Applicability Statements

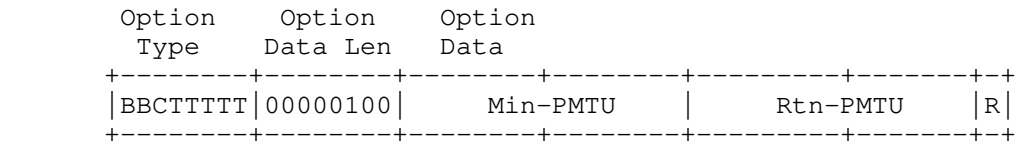
This Hop-by-Hop Option header is intended to be used in environments such as Data Centers and on paths between Data Centers, to allow them to better take advantage of a path that is able to support a large PMTU. For example, it helps inform a sender that the path includes links that have a MTU of 9,000 Bytes. This has many performance advantages compared to the current practice of limiting packets to 1280 Bytes.

The design of the option is sufficiently simple that it could be executed on a router's fast path. To create critical mass for this to happen will have to be a strong pull from router vendors customers. This could be the case for connections within and between Data Centers.

The method could also be useful in other environments, including the general Internet.

5. IPv6 Minimum Path MTU Hop-by-Hop Option

The Minimum Path MTU Hop-by-Hop Option has the following format:



Option Type:

BB 00 Skip over this option and continue processing.

C 1 Option data can change en route to the packet's final destination.

TTTT 11110 Experimental Option Type from [IANA-HBH].

Length: 4 The size of the each value field in Option Data field supports Path MTU values from 0 to 65,535 octets.

Min-PMTU: n 16-bits. The minimum PMTU in octets, reflecting the smallest link MTU that the packet experienced across the path. This is called the Reported PMTU. A value less than the IPv6 minimum link MTU [RFC8200] should be ignored.

Rtn-PMTU: n 15-bits. The returned mimimum PMTU, carrying the 15 most significant bits of the latest received Min-PMTU field. The value zero means that no Reported MTU is being returned.

R n 1-bit. R-Flag. Set by the source to signal that the destination should include the received Reported PMTU in Rtn-PMTU field.

NOTE: The encoding of the final two octets (Rtn-PMTU and R-Flag) could be implemented by a mask of the latest received Min-MTU value with 0xFFFE, discarding the right-most bit and then performing a logical 'OR' with the R-Flag value of the sender.

6. Router, Host, and Transport Behaviors

6.1. Router Behaviour

Routers that do not support Hop-by-Hop options SHOULD ignore this option and SHOULD forward the packet.

Routers that support Hop-by-Hop Options, but do not recognize this option SHOULD ignore the option and SHOULD forward the packet.

Routers that recognize this option SHOULD compare the Reported PMTU in the Min-PMTU field and the MTU configured for the outgoing link. If the MTU of the outgoing link is less than the Reported PMTU, the router rewrites the Reported PMTU in the Option to use the smaller value.

The router MUST ignore and not change the Rtn-PMTU field and R-Flag in the option.

Discussion:

- The design of this Hop-by-Hop Option makes it feasible to be implemented within the fast path of a router, because the required processing is simple.

6.2. Host Behavior

The source host that supports this option SHOULD create a packet with this Hop-by-Hop Option and fill the Min-PMTU field of the option with the MTU of configured for the link over which it will send the packet on the next hop towards the destination.

The source host may request that the destination host return the received minimum MTU value by setting the R-Flag in the option. This will cause the destination host to include a PMTU option in an outgoing packet.

Discussion:

- This option does not need to be sent in all packets belonging to a flow. A transport protocol (or packetization layer) can set this option only on specific packets used to test the path.

- In the case of TCP, the option could be included in packets carrying a SYN segment as part of the connection set up, or can periodically be sent in packets carrying other segments. Including this packet in a SYN could increase the probability that SYN segment is lost, when routers on the path drop packets with this option. Including this option in a large packet is not likely to be useful, since the large packet might itself also be dropped by a link along the path with a smaller MTU, preventing the Reported PMTU information from reaching the Destination node.

- The use with datagram transport protocols (e.g. UDP) is harder to characterize because applications using datagram transports range from very short-lived (low data-volume applications) exchanges, to longer (bulk) exchanges of packets between the Source and Destination nodes [RFC8085].

- For applications that use Anycast, this option should be included in all packets as the actual destination will vary due to the nature of Anycast.

- Simple-exchange protocols (i.e low data-volume applications [RFC8085] that only send one or a few packets per transaction, could be optimized by assuming that the Path MTU is symmetrical, that is where the Path MTU is the same in both directions, or at least not smaller in the return path. This optimisation does not hold when the paths are not symmetric.

- The use of this option with DNS and DNSSEC over UDP ought to work as long as the paths are symmetric. The DNS server will learn the Path MTU from the DNS query messages. If the return Path MTU is smaller, then the large DNSSEC response may be dropped and the known problems with PMTUD will occur. DNS and DNSSEC over transport protocols that can carry the Path MTU should work.

The Source Host can request the destination host to send a packet carrying the PMTU Option using the R-Flag.

A Destination Host SHOULD respond to each packet received with the R-Flag set, by setting the PMTU Option in the next packet that it sends to the Source Host by the same upper layer protocol instance.

The upper layer protocol MAY generate a packet when any of these conditions is met when the R Flag is set in the PMTU Option and either:

- It is the first Reported PMTU value it has received from the Source.

- The Reported PMTU value is lower than previously received.

The R-Flag SHOULD NOT be set when the PMTU Option was sent solely to carry the feedback of a Reported PMTU.

The PMTU Option sent back to the source SHOULD contain the outgoing link MTU in Min-PMTU field and SHOULD set the last Received PMTU in the Rtn-PMTU field. If these values are not present the field MUST be set to zero.

For a connection-oriented upper layer protocol, this could be implemented by saving the value of the last received option within the connection context. This last received value is then used to set the return Path MTU field for all packets belonging to this flow that carry the IPv6 Minimum Path MTU Hop-by-Hop Option.

A connection-less protocol, e.g., based on UDP, requires the application to be updated to cache the Received PMTU value, and to ensure that this corresponding value is used to set the last Received PMTU in the Rtn-PMTU field of any PMTU Option that it sends.

NOTE: The Rtn-PMTU value is specific to the instance of the upper layer protocol (i.e. matching the IPv6 flow ID, port-fields in UDP or the SPI in IPsec, etc), not the protocol itself, because network devices can make forwarding decisions that impact the PMTU based on the presence and values of these upper layer fields, and therefore these fields need to correspond to those of the packets for the flow received by the Destination Host set to ensure feedback is provided to the corresponding Source Host.

NOTE: An upper layer protocol that send packets from the Destination Host towards the Source Host less frequently than the Destination Host receives packets from the Source Host, provides less frequent feedback of the received Min-PMTU value. However, it will always needs to send the most recent value.

Discussion:

- A simple mechanism could only send an MTU Option with the Rtn-PMTU field filled in the first time this option is received or when the Received PMTU is reduced. This is good because it limits the number sent, but there is no provision for retransmission of the PMTU Option fails to reach the sender, or the sender looses state.

- The Reported PMTU value could increase or decrease over time. For instance, it would increase when the path changes and the packets become then forwarded over a link with a MTU larger than the link previously used.

6.3. Transport Behavior

A transport endpoint using this option needs to use a method to verify the information provided by this option.

The Received PMTU does not necessarily reflect the actual PMTU between the sender and destination. Care therefore needs to be exercised in using this value at the sender. Specifically:

- If the Received PMTU value returned by the Destination is the same as the initial Reported PMTU value, there could still be a router or layer 2 device on the path that does not support this PMTU. The usable PMTU therefore needs to be confirmed.

- If the Received PMTU value returned by the Destination is smaller than the initial Reported PMTU value, this is an indication that there is at least one router in the path with a smaller MTU. There could still be another router or layer 2 device on the path that does not support this MTU.

- If the Received PMTU value returned by the Destination is larger than the initial Reported PMTU value, this may be a corrupted, delayed or mis-ordered response, and SHOULD be ignored.

A sender needs to discriminate between the Received PMTU value in a PTB message generated in response to a Hop-by-Hop option requesting this, and a PTB message received from a router on the path.

A PMTUD or PLPMTUD method could use the Received PMTU value as an initial target size to probe the path. This can significantly decrease the number of probe attempts (and hence time taken) to arrive at a workable PMTU. It has the potential to complete discovery of the correct value in a single Round Trip Time (RTT), even over paths that may have successive links configured with lower MTUs.

Since the method can delay notification of an increase in the actual PMTU, a sender with a link MTU larger than the current PMTU SHOULD periodically probe for a PMTU value that is larger than the Received PMTU value. This specification does not define an interval for the time between probes.

Since the option consumes less capacity than an a full probe packet, there may be advantage in using this to detect a change in the path characteristics.

NOTE: Further details to be included in next version.

NOTE: A future version of the document will consider more the impact of Equal Cost Multipath (ECMP). Specifically, whether a Received PMTU value should be maintained by the method for each transport endpoint, or for each network address, and how these are best used by methods such as PLPMTUD or DPLPMTUD.

7. IANA Considerations

No IANA assignments are requested. Document uses experimental option from [IANA-HBH].

8. Security Considerations

The method has no way to protect the destination from off-path attack using this option in packets that do not originate from the source. This attack could be used to inflate or reduce the size of the reported PMTU. Mechanisms to provide this protection can be provided at a higher layer (e.g., the transport packetization layer using PLPMTUD or DPLPMTUD), where more information is available about the size of packet that has successfully traversed a path.

The method solicits a response from the destination, which should be used to generate a response to the IPv6 node originating the option packet. A malicious attacker could generate a packet to the destination for a previously inactive flow or one that advertises a change in the size of the MTU for an active flow. This would create additional work at the destination, and could induce creation of state when a new flow is created. It could potentially result in additional traffic on the return path to the sender, which could be mitigated by limiting the rate at which responses are generated.

A sender MUST check the quoted packet within the PTB message to validate that the message is in response to a packet that was originated by the sender. This is intended to provide protection against off-path insertion of ICMP PTB messages by an attacker trying to disrupt the service. Messages that fail this check MAY be logged, but the information they contain MUST be discarded.

TBD

9. Acknowledgments

A somewhat similar mechanism was proposed for IPv4 in 1988 in [RFC1063] by Jeff Mogul, C. Kent, Craig Partridge, and Keith McCloghire. It was later obsoleted in 1990 by [RFC1191] the current deployed approach to Path MTU Discovery.

Helpful comments were received from Tom Herbert, Tom Jones, Fred Templin, Ole Troan, [Your name here], and other members of the 6MAN working group.

10. Change log [RFC Editor: Please remove]

draft-hinden-6man-mtu-option-02, 2019-July-5

- Changed option format to also include the Returned MTU value and Return flag and made related text changes in Section 6.2 to describe this behaviour.

- ICMP Packet Too Big messages are no longer used for feedback to the Source host.
- Added to Acknowledgements Section that a similar mechanism was proposed for IPv4 in 1988 in [RFC1063].
- Editorial changes.

draft-hinden-6man-mtu-option-01, 2019-March-05

- Changed requested status from Standards Track to Experimental to allow use of experimental option type (11110) to allow for experimentation. Removed request for IANA Option assignment.
- Added Section 2 "Motivation and Problem Solved" section to better describe what the purpose of this document is.
- Added Appendix A describing planned experiments and how the results will be measured.
- Editorial changes.

draft-hinden-6man-mtu-option-00, 2018-Oct-16

- Initial draft.

11. References

11.1. Normative References

[IANA-HBH] "Destination Options and Hop-by-Hop Options", <https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2>.

[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>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

11.2. Informative References

[RFC1063] Mogul, J., Kent, C., Partridge, C., and K. McCloghrie, "IP MTU discovery options", RFC 1063, DOI 10.17487/RFC1063, July 1988, <https://www.rfc-editor.org/info/rfc1063>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

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

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

Appendix A. Planned Experiments

TBD

This section will describe a set of experiments planned for the use of the option defined in this document. There are many aspects of the design that require experimental data or experience to evaluate this experimental specification.

This includes experiments to understand the pathology of packets sent with the specified option to determine the likelihood that they are lost within specific types of network segment.

This includes consideration of the cost and alternatives for providing the feedback required by the mechanism and how to effectively limit the rate of transmission.

This includes consideration of the potential for integration in frameworks such as that offered by DPLPMTUD.

There are also security-related topics to be understood as described in the Security Considerations (Section 8).

Authors' Addresses

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
USA

Email: bob.hinden@gmail.com

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

Email: gorry@erg.abdn.ac.uk

Network Working Group
Internet-Draft
Updates: 4861, 5175 (if approved)
Intended status: Standards Track
Expires: September 8, 2019

R. Hinden
Check Point Software
B. Carpenter
Univ. of Auckland
B. Zeeb

March 7, 2019

IPv6 Router Advertisement IPv6-Only Flag
draft-ietf-6man-ipv6only-flag-05

Abstract

This document specifies a Router Advertisement Flag to indicate to hosts that the administrator has configured the router to advertise that the link is IPv6-Only. This document updates RFC4861 and RFC5175.

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 8, 2019.

Table of Contents

1. Introduction

This document specifies a Router Advertisement Flag to indicate to hosts that the administrator has configured the router to advertise that the link is IPv6-Only. The flag only applies to IPv6 default routers.

Hosts that support IPv4 and IPv6, usually called dual stack hosts, need to also work efficiently on IPv6-Only links, i.e, links where there are no IPv4 routers and/or IPv4 services. Dual stack is the default configuration for most current host operating systems such as Windows 10, iOS, Android, Linux, and BSD, as well as devices such as some printers. Monitoring of an IPv6-Only link, for example at the IETF 100 meeting in Singapore, shows that current dual stack hosts will create local auto-configured IPv4 addresses and attempt to reach IPv4 services, even though they cannot configure a normal address using DHCP. This may be a problem for several reasons, depending on the equipment in use and its configuration, especially on large wireless networks:

- It may result in an undesirable level of wasted Layer 2 broadcast traffic.

- Switches in multi-segment wireless networks may create IPv4 state for dual stack hosts (in particular, ARP cache entries to support ARP proxying).

- Such traffic may drain battery power on wireless hosts that have no interest in link-local IPv4, ARP, and DHCPv4 relay traffic, but receive unwanted IPv4 packets. [RFC7772] indicates how this risk might be quantified.

- Similarly, hosts may waste battery power on futile attempts to access services by sending IPv4 packets.

- On an IPv6-Only link, IPv4 might be used for malicious purposes and pass unnoticed by IPv6-Only monitoring mechanisms.

In networks with managed infrastructure whose equipment allows it, these problems could be mitigated by configuring the Layer 2 infrastructure to drop IPv4 and ARP traffic by filtering Ethertypes 0x0800 and 0x0806 [IANA-Ethertype]. IPv6 uses a different Ethertype, 0x86DD, so this filtering will not interfere with IPv6 traffic. Depending on the equipment details, this would limit the traffic to the link from an IPv4 sender to the switch, and would drop all IPv4 and ARP broadcast packets at the switch. This document recommends using such mechanisms when available.

However, hosts transmitting IPv4 packets would still do so, consuming their own battery power and some radio bandwidth. The intent of this specification is to provide a mechanism that prevents such traffic, and also works on networks without the ability to filter L2 traffic, or where there are portions of a network without the ability to filter L2 traffic. It may also be valuable on unmanaged networks using routers pre-configured for IPv6-Only operations and where Layer 2 filtering is unavailable.

An assumption of this document is that because it is an IPv6-Only link there is no IPv4 DHCP server or relay active on the link. This further means that the DHCP option to disable IPv4 stateless auto-configuration [RFC2563] can not be used.

The remainder of this document therefore assumes that neither effective Layer 2 filtering nor the RFC 2563 DHCP option is applicable to the link concerned.

Because there is no IPv4 support on an IPv6-Only link, the only way to notify the dual stack hosts that this link is IPv6-Only is to use an IPv6 mechanism. An active notification will be much more precise than attempting to deduce this fact by the lack of IPv4 responses or traffic.

This document therefore defines a mechanism that a router administrator can use to inform hosts that this is an IPv6-Only link on their default routers such that they can disable IPv4 on this link, mitigating all of the above problems. The mechanism is based on the IPv6 Router Advertisement message because this is a type of message that is certain to be received by every dual stack host, regardless of what network management protocols may or may not be in use.

IPv4-only hosts, and dual-stack hosts that do not recognize the new flag, may continue to attempt IPv4 operations, in particular IPv4 discovery protocols typically sent as link-layer broadcasts. This legacy traffic cannot be prevented by any IPv6 mechanism. The value of the new flag is limited to hosts that recognize it.

A possible subsidiary use of the IPv6-Only flag is using it to trigger IPv6-Only testing and validation on a link.

This document specifies a new flag for Router Advertisement Flag [RFC5175]. It updates [RFC5175] to add this flag. It also updates [RFC4861] to add an additional item to check and report.

2. Requirements Language

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

3. Applicability Statements

This OPTIONAL mechanism is designed to allow administrators to notify hosts that the link is IPv6-Only. It SHOULD be only used in IPv6-Only links (see Section 4 for definition of IPv6-Only). For a VLAN, the IPv6-Only flag only applies to the specific VLAN on which it was received.

Dual stack hosts that have IPv4 active configuration obtained from the network (e.g., via DHCP), can ignore the flag and continue to use IPv4.

Administrators MUST only use this mechanism if they are certain that the link is IPv6-Only. For example, in cases where there is a need to continue to use IPv4, when there are intended to be IPv4-only hosts or IPv4 routers on the link, setting this flag to 1 is a configuration error.

This mechanism is intended to be compatible with link-layer solutions that filter out IPv4 traffic.

4. IPv6-Only Definition

IPv6-Only is defined to mean that no other versions of Internet Protocol than IPv6 are intentionally in use directly on the link. Today this effectively simply means that IPv4 is not intentionally in use on the link, and it includes:

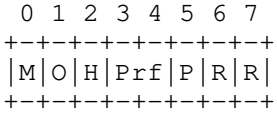
* No IPv4 traffic on the link.
* No IPv4 routers on the link.
* No DHCPv4 servers on the link.
* No IPv4 accessible services on the link.
* All IPv4 and ARP traffic may be blocked at Layer 2 by the administrator.

It is expected that on IPv6-Only networks it will be common to access to IPv4 external services by techniques such as NAT64 [RFC6146] and DNS64 [RFC6147] at the edge of the network. This is beyond the scope of this document.

Note that IPv6-Only provides no information about other network protocols than IP (and ARP) in use directly over the link layer. It is out of scope of this specification whether any such protocol is in use on the link or whether any protocol is tunneled over IPv6.

5. IPv6-Only Flag

RFC5175 currently defines the flags in the NDP Router Advertisement message and these flags are registered in the IANA IPv6 ND Router Advertisement flags Registry [IANA-RF]. This currently contains the following one-bit flags defined in published RFCs:



M	Managed Address Configuration Flag [RFC4861]
O	Other Configuration Flag [RFC4861]
H	Mobile IPv6 Home Agent Flag [RFC3775]
Prf	Router Selection Preferences [RFC4191]
P	Neighbor Discovery Proxy Flag [RFC4389]

R Reserved

This document defines bit 6 to be the IPv6-Only Flag:

S IPv6-Only Flag

This flag has two values. These are:

0 This is not an IPv6-Only link
1 This is an IPv6-Only link

RFC 5175 requires that unused flag bits be set to zero. Therefore, a router that does not support the new flag will not appear to assert that this is an IPv6-Only link.

Hosts receiving the Router Advertisement SHOULD only process this flag if the advertising router is a Default Router. Specifically, if the Lifetime field in the Router Advertisement is not zero, otherwise it SHOULD be ignored. This is done to allow some IPv6 routers to advertise information without being a Default Router and providing IPv6 connectivity.

Note that although this mechanism uses one of only two reserved flag bits in the RA, an extension mechanism is defined in Section 4 of [RFC5175] in case additional flags are ever required for future extensions. It should be noted that since RFC5175 was published in 2008, no new RA flags have been assigned in the IANA registry.

6. Router and Operational Considerations

Default IPv6 routers that are on an IPv6-Only link SHOULD be configured by the administrator to set the IPv6-Only flag to 1 on interfaces on this link. In all other cases the flag SHOULD NOT be set to 1.

The intent is that the administrator of the router configures the router to set the IPv6-Only flag if and only if she/he wants to tell the hosts on the link that the link is IPv6-Only. This is a configuration flag, it is not something that the router decides on its own. Routers MAY log a configuration error if the flag is set and IPv4 is still active on the router's interface to the link.

Routers implementing this document SHOULD log to system or network management inconsistent setting of the IPv6-Only flag. This extends the behaviour specified in Section 6.2.7 of [RFC4861].

Operators of large IPv6-Only wireless links are advised to also use Layer 2 techniques to drop IPv4 and ARP packets (Ethertypes 0x0800 and 0x0806) at all switches, and to ensure that IPv4 and ARP features are disabled in all switches.

7. Host Behavior Considerations

Hosts that support the IPv6-Only RA flag MUST have a configuration option to ignore or process the flag. The motivation for this configuration option is for hosts that are capable of processing the IPv6-Only flag to only act on the flag if they are configured to do so.

If there are multiple IPv6 default routers on a link, they might send different values of the flag. If at least one IPv6 default router sends the flag with value 0, a dual stack host MUST NOT assume that the link is IPv6-Only. If all IPv6 default routers send the flag with value 1, a dual stack host SHOULD assume that this is an IPv6-Only link.

A host that receives only RAs with the flag set to 1 SHOULD NOT attempt any IPv4 operations, unless it subsequently receives at least one RA with the flag set to zero. As soon as such an RA is received, IPv4 operations MAY be started.

If the host has active IPv4 configuration information obtained from the network (e.g., via DHCP), the flag can be ignored and IPv4 operations can continue. The host MAY implement a policy overriding these default behaviors.

In the event that the host subsequently receives at least one RA with the flag set to zero IPv4 operations MAY be started.

A host MAY delay all IPv4 operations at start-up or reconnection until a reasonable time has elapsed for RA messages to arrive. If all RAs received have the flag set to 1, a host SHOULD NOT attempt IPv4 operations.

In all of the above, the flag's value is considered valid for the lifetime of the default router concerned, unless a subsequent RA delivers a different flag value. If a default router expires (i.e., no RA is received that refreshes its lifetime), the host must remove this router's flag value from consideration. If the result is that all surviving default routers have the flag set to 1, the host SHOULD

assume that the link is IPv6-Only. In other words, at any given time, the state of the flag as seen by the host is the logical AND of the flags sent by all unexpired default IPv6 routers on the link.

This also means that if all default routers on the link have set the flag, the resulting host state for the link is IPv6-Only. If the lifetimes of all the routers on the link subsequently expire, then the host state for the link is not IPv6-Only.

8. IANA Considerations

IANA is requested to assign the new Router Advertisement flag defined in Section 5 of this document. Bit 6 is the next available bit in this registry, IANA is requested to use this bit unless there is a reason to use another bit in this registry.

IANA is also requested to register this new flag bit in the IANA IPv6 ND Router Advertisement flags Registry [IANA-RF].

9. Security Considerations

This document shares the security issues with other parts of IPv6 Neighbor Discovery. [RFC6104] discusses certain attacks and mitigations. General techniques to protect Router Advertisement traffic such as Router Guard [RFC6105] are useful in protecting against these vulnerabilities.

A bad actor could use this mechanism to attempt turn off IPv4 service on a link that is intentionally using IPv4, by sending Router Advertisements with the IPv6-Only flag set to 1. There are several protections to reduce this attack. These are:

- There is configuration setting that controls if the host should process the IPv6-Only flag. This gives local control over using the flag and reduces the ability of a bad actor to turn off IPv4 for hosts that support the flag.

- As long as there are one or more routers sending Router Advertisements with this flag set to 0, they would override this attack given the mechanism in Section 5. Specifically a host would only turn off IPv4 service if it wasn't hearing any Router Advertisement with the flag set to 0. If the advice in Section 6 is followed, this attack will fail.

- An attack would not succeed if the dual stack hosts had active IPv4 configuration. As specified in Section 7, a dual stack host will ignore the flag if it has active IPv4 configuration.

In a situation where the bad actor has control of all routers on the link and sends Router Advertisements with the IPv6-Only flag set to 1 from all of them and if the hosts don't have assigned IPv4 addresses, the attack will succeed, but so will many other forms of router-based attack.

Conversely, a bad actor could use this mechanism to turn on, or pretend to turn on, IPv4 service on an IPv6-Only link, by sending Router Advertisements with the flag set to 0. However, this is really no different than what such a bad actor can do anyway, if they have the ability to configure a bogus router in the first place. The advice in Section 6 will minimize such an attack by limiting it to a single link.

Note that manipulating the Router Preference [RFC4191] will not affect either of these attacks: any IPv6-Only flag of 0 will always override all flags set to 1.

The new flag is neutral from an IPv6 privacy viewpoint, since it does not affect IPv6 operations in any way. From an IPv4 privacy viewpoint, it has the potential benefit of suppressing unnecessary traffic that might reveal the existence of a host and the correlation between its hardware and IPv4 addresses. It should be noted that hosts that don't support this flag are not protected from IPv4-based attacks.

10. Acknowledgments

A closely related proposal was published earlier as [I-D.ietf-sunset4-noipv4].

Helpful comments were received from Lorenzo Colitti, David Farmer, Fernando Gont, Nick Hilliard, Lee Howard, Erik Kline, Jen Linkova, Veronika McKillop, George Michaelson, Alexandre Petrescu, Michael Richardson, Mark Smith, Barbara Stark, Tatuya Jinmei, Ole Troan, James Woodyatt, and other members of the 6MAN working group.

Bjoern Zeeb has also produced a variant of this proposal and proposed an IPv6 transition plan in [I-D.bz-v4goawayflag].

11. Change log [RFC Editor: Please remove]

draft-ietf-6man-ipv6only-flag-05, 2019-March-7:

* Added a host configuration option to Section 7 that controls if the host should process the IPv6-Only flag. This provides local control over using the use of flag and reduces the

ability of a bad actor to turn off IPv4 for hosts that support the flag.
* Changed Section 7 to specify that the host can ignore flag set to 1 if it has active IPv4 configuration obtained from the network (e.g., via DHCP). Similar changes to Section 3 and Section 9
* Clarification in Section 6 to strengthen the text about the administrators intent.
* Added Bjoern Zeeb as an author.
* Updated information on FreeBSD implementation in Appendix A.1
* Editorial changes.

draft-ietf-6man-ipv6only-flag-04, 2018-November-4:

* Added text to Section 1 explaining why the mechanism is based on Router Advertisements.
* Added text to Section 3 that for a VLAN, the IPv6-Only flag only applies to the specific VLAN on which it was received.
* Changed Section 3 that administrators MUST only use this mechanism if they are certain that the link is IPv6-Only, instead of SHOULD.
* Added ARP to Section 4 protocols that the IPv6-Only flag applies to.
* Renamed the IPv6-Only flag label from "6" to "S".
* Added pointers to Section 7.2.7 of RFC4861 in Section 6.
* Added that RFC4861 is also updated by Section 6 for routers implementing this flag.
* Changed Section 7 from SHOULD NOT to MUST NOT.
* Added Appendix A on implementations and testing.
* Many small clarifications based on IPv6 list discussion and editorial changes.

draft-ietf-6man-ipv6only-flag-03, 2018-October-16:

* Reorganized text about problem statement and applicability
* Added note about shortage of flag bits
* Clarified text about logging configuration error in Section 6
* Editorial changes.

draft-ietf-6man-ipv6only-flag-02, 2018-August-14:

* Added text to Section 9 to clarify that hosts not supporting this flag are not protected from IPv4-based attacks.
* Editorial changes.

draft-ietf-6man-ipv6only-flag-01, 2018-June-29:

* Added text to section that defines what IPv6-Only includes to clarify that only other version of the Internet Protocol are in scope.
* Added clarification if the lifetime of all routers expire.
* Editorial changes.

draft-ietf-6man-ipv6only-flag-00, 2018-May-21:

* Changed the file name to draft-ietf-6man-ipv6only-flag to match the current tile and that it is a w.g. draft.
* Added new section that defines what IPv6-Only includes.
* Expanded description of using Layer 2 filter to block IPv4 and ARP traffic.
* Editorial changes.

draft-hinden-ipv4flag-04, 2018-April-16:

* Changed the name of the document and flag to be the IPv6-Only flag.
* Rewrote text to make it affirmative that this is used by an administrator to tell the hosts that the link is IPv6-Only.
* Added an Applicability Statements section to scope the intend use.
* Changed requirement language to upper case, added Requirements Language section with references to [RFC2119] and [RFC8174].
* Editorial changes.

draft-hinden-ipv4flag-03, 2018-Feb-15:

* Changed terminology to use "link" instead of "network".
* Improved text in Section 4. "Host Behavior Considerations" and added suggestion to only perform IPv4 if an application requests it.
* Added clarification that the bit is set because an administrator configured the router to send it.
* Editorial changes.

draft-hinden-ipv4flag-02, 2018-Feb-15:

* Improved text in introduction.
* Added reference to current IANA registry in Section 2.

* Editorial changes.

draft-hinden-ipv4flag-01, 2017-Dec-12

* Inverted name of flag from "Available" to "Unavailable".
* Added problem description and clarified scope.
* Added router and operational considerations.
* Added host behavior considerations.
* Extended security considerations.
* Added Acknowledgment section, including reference to prior sunset4 draft.

draft-hinden-ipv4flag-00, 2017-Nov-17:

* Original version.

12. References

12.1. Normative References

[IANA-Ethertype] "Ether Types", <https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml#ieee-802-numbers-1>.

[IANA-RF] "IPv6 ND Router Advertisement flags", <https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml#icmpv6-parameters-11>.

[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>.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <https://www.rfc-editor.org/info/rfc4191>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC5175] Haberman, B., Ed. and R. Hinden, "IPv6 Router Advertisement Flags Option", RFC 5175, DOI 10.17487/RFC5175, March 2008, <https://www.rfc-editor.org/info/rfc5175>.

[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>.

12.2. Informative References

[I-D.bz-v4goawayflag]
Zeeb, B., "IPv6 Router Advertisement IPv4 GoAway Flag", draft-bz-v4goawayflag-00 (work in progress), March 2018.

[I-D.ietf-sunset4-noipv4]
Perreault, S., George, W., Tsou, T., Yang, T., and J. Tremblay, "Turning off IPv4 Using DHCPv6 or Router Advertisements", draft-ietf-sunset4-noipv4-01 (work in progress), December 2014.

[RFC2563] Troll, R., "DHCP Option to Disable Stateless Auto-Configuration in IPv4 Clients", RFC 2563, DOI 10.17487/RFC2563, May 1999, <https://www.rfc-editor.org/info/rfc2563>.

[RFC6104] Chown, T. and S. Venaas, "Rogue IPv6 Router Advertisement Problem Statement", RFC 6104, DOI 10.17487/RFC6104, February 2011, <https://www.rfc-editor.org/info/rfc6104>.

[RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <https://www.rfc-editor.org/info/rfc6105>.

[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>.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <https://www.rfc-editor.org/info/rfc6147>.

[RFC7772] Yourtchenko, A. and L. Colitti, "Reducing Energy Consumption of Router Advertisements", BCP 202, RFC 7772, DOI 10.17487/RFC7772, February 2016, <https://www.rfc-editor.org/info/rfc7772>.

[Scapy_RA]
"Router Advertisements with scapy (NETLAB)",
<https://samsclass.info/124/proj11/proj9xN-scapy-ra.html>.

Appendix A. Implementaton Status [RFC Editor: Please remove]

At the time this document was written there is one implementation and a few comparability tests.

A.1. FreeBSD Implementation

A FreeBSD implementation was written by Bjoern A. Zeeb.

Summary:

Changes for the IPv6-Only flag include updates of user space utilities to announce the "S" (IPv6-Only) flag to the network and to show it in management utilities.

The kernel logic includes a global flag to disable processing of the IPv6-Only flag even if the logic to act upon the IPv6-Only flag is compiled in. There are checks for IPv4 configuration on a receiving interface, which if found, will also force the IPv6-Only flag to be ignored.

Further there are input and output filters for IPv4, ARP, and REVARP in place for when the flag passes the aforementioned checks and is enabled.

In addition to the draft there is a manual option to enable the IPv6-Only filtering logic manually to observe the system behaviour on links without router(s) advertising the IPv6-Only flag.

The code was tested with 2 FreeBSD IPv6 routers, a FreeBSD laptop on ethernet as well as wifi, and with Win10 and OSX clients (which did not fall over with the "S" flag set but not understood).

More information and updates can be found at:

https://wiki.freebsd.org/IPv6/IPv6OnlyRAFlag

A.2. Test using Scapy

Independent tests have been done using [Scapy_RA] by Alexandre Petrescu and Brian Carpenter to verify that setting the IPv6-Only

Flag did not break legacy hosts. Both verified that setting this flag did not cause any adverse effects on Windows 10 and Android.

Authors' Addresses

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
USA

Email: bob.hinden@gmail.com

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Bjoern A. Zeeb
Bruehlstr. 19
Bondorf 71149
Germany

Email: bzeeb+ietf@zabbadoz.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2020

C. Filsfils, Ed.
D. Dukes, Ed.
Cisco Systems, Inc.
S. Previdi
Huawei
J. Leddy
Individual
S. Matsushima
Softbank
D. Voyer
Bell Canada

October 22, 2019

IPv6 Segment Routing Header (SRH)
draft-ietf-6man-segment-routing-header-26

Abstract

Segment Routing can be applied to the IPv6 data plane using a new type of Routing Extension Header called the Segment Routing Header. This document describes the Segment Routing Header and how it is used by Segment Routing capable nodes.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 24, 2020.

Table of Contents

1. Introduction

Segment Routing can be applied to the IPv6 data plane using a new type of Routing Header called the Segment Routing Header. This document describes the Segment Routing Header and how it is used by Segment Routing capable nodes.

The Segment Routing Architecture [RFC8402] describes Segment Routing and its instantiation in two data planes; MPLS and IPv6.

The encoding of IPv6 segments in the Segment Routing Header is defined in this document.

This document uses the terms Segment Routing, SR Domain, SRv6, Segment ID (SID), SRv6 SID, Active Segment, and SR Policy as defined in [RFC8402].

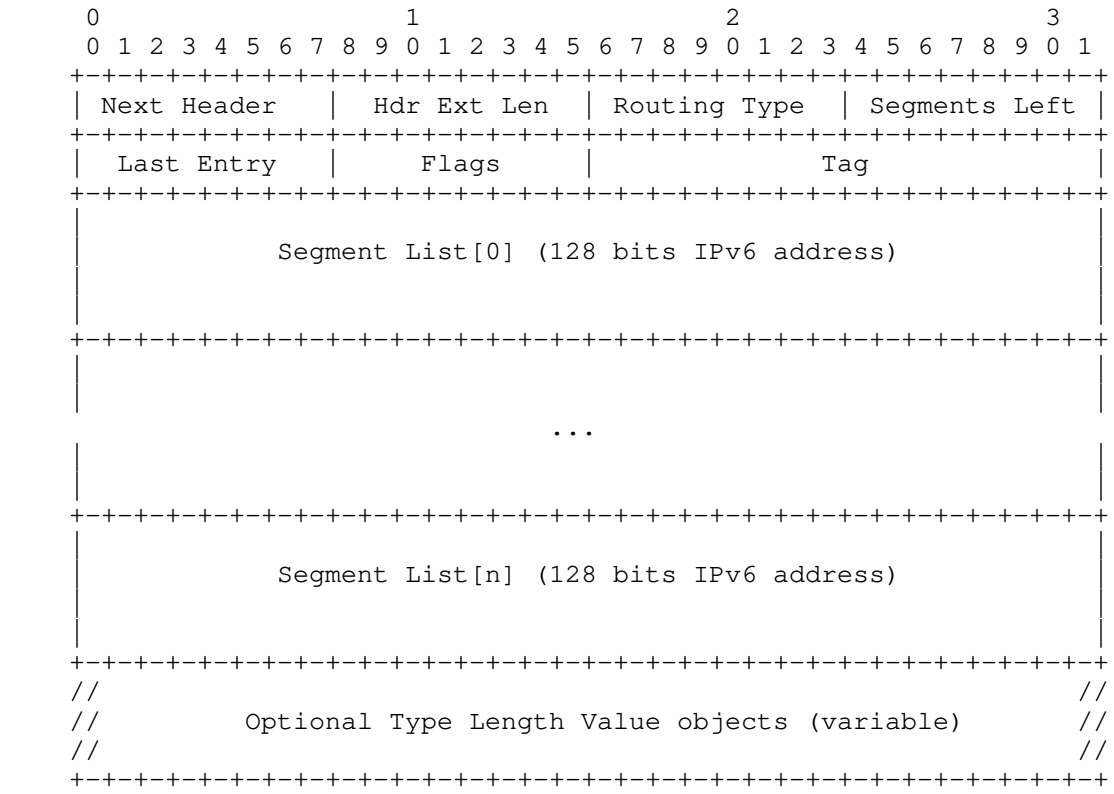
1.1. Requirements Language

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

2. Segment Routing Header

Routing Headers are defined in [RFC8200]. The Segment Routing Header has a new Routing Type (suggested value 4) to be assigned by IANA.

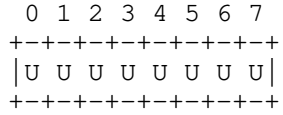
The Segment Routing Header (SRH) is defined as follows:



where:

- Next Header: Defined in [RFC8200] Section 4.4
- Hdr Ext Len: Defined in [RFC8200] Section 4.4
- Routing Type: TBD, to be assigned by IANA (suggested value: 4).
- Segments Left: Defined in [RFC8200] Section 4.4
- Last Entry: contains the index (zero based), in the Segment List, of the last element of the Segment List.

- Flags: 8 bits of flags. Section 8.1 creates an IANA registry for new flags to be defined. The following flags are defined:



  U: Unused and for future use. MUST be 0 on transmission and ignored on receipt.

- Tag: tag a packet as part of a class or group of packets, e.g., packets sharing the same set of properties. When tag is not used at source it MUST be set to zero on transmission. When tag is not used during SRH Processing it SHOULD be ignored. Tag is not used when processing the SID defined in Section 4.3.1. It may be used when processing other SIDs that are not defined in this document. The allocation and use of tag is outside the scope of this document.

- Segment List[n]: 128 bit IPv6 addresses representing the nth segment in the Segment List. The Segment List is encoded starting from the last segment of the SR Policy. I.e., the first element of the segment list (Segment List [0]) contains the last segment of the SR Policy, the second element contains the penultimate segment of the SR Policy and so on.

- Type Length Value (TLV) are described in Section 2.1.

In the SRH, the Next Header, Hdr Ext Len, Routing Type, and Segments Left fields are defined in Section 4.4 of [RFC8200]. Based on the constraints in that section, Next Header, Header Ext Len, and Routing Type are not mutable while Segments Left is mutable.

The mutability of the TLV value is defined by the most significant bit in the type, as specified in Section 2.1.

Section 4.3 defines the mutability of the remaining fields in the SRH (Flags, Tag, Segment List) in the context of the SID defined in this document.

New SIDs defined in the future MUST specify the mutability properties of the Flags, Tag, and Segment List and indicate how the HMAC TLV (Section 2.1.2) verification works. Note, that in effect these fields are mutable.

Consistent with the source routing model, the source of the SRH always knows how to set the segment list, Flags, Tag and TLVs of the SRH for use within the SR Domain. How it achieves this is outside the scope of this document, but may be based on topology, available SIDs and their mutability properties, the SRH mutability requirements of the destination, or any other information.

2.1. SRH TLVs

This section defines TLVs of the Segment Routing Header.

A TLV provides meta-data for segment processing. The only TLVs defined in this document are the HMAC (Section 2.1.2) and PAD (Section 2.1.1) TLVs. While processing the SID defined in Section 4.3.1, all TLVs are ignored unless local configuration indicates otherwise (Section 4.3.1.1.1). Thus, TLV and HMAC support is optional for any implementation, however, an implementation adding or parsing TLVs MUST support PAD TLVs. Other documents may define additional TLVs and processing rules for them.

TLVs are present when the Hdr Ext Len is greater than (Last Entry+1)*2.

While processing TLVs at a segment endpoint, TLVs MUST be fully contained within the SRH as determined by the Hdr Ext Len. Detection of TLVs exceeding the boundary of the SRH Hdr Ext Len results in an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Hdr Ext Len field of the SRH, and the packet being discarded.

An implementation MAY limit the number and/or length of TLVs it processes based on local configuration. It MAY:

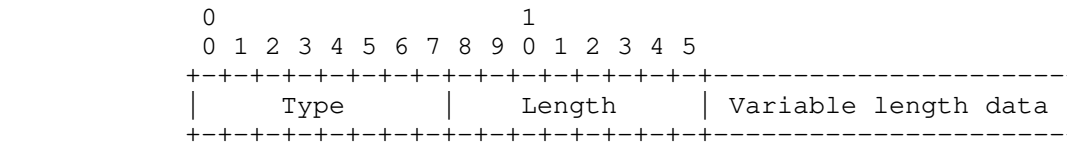
- Limit the number of consecutive Pad1 (Section 2.1.1.1) options to 1. If padding of more than one byte is required, then PadN (Section 2.1.1.2) should be used.

- Limit the length in PadN to 5.

- Limit the maximum number of non-Pad TLVs to be processed.

- Limit the maximum length of all TLVs to be processed.

The implementation MAY stop processing additional TLVs in the SRH when these configured limits are exceeded.



Type: An 8 bit codepoint from Segment Routing Header TLVs Registry TBD IANA Reference. Unrecognized Types MUST be ignored on receipt.

Length: The length of the Variable length data in bytes.

Variable length data: Length bytes of data that is specific to the Type.

Type Length Value (TLV) entries contain OPTIONAL information that may be used by the node identified in the Destination Address (DA) of the packet.

Each TLV has its own length, format and semantic. The codepoint allocated (by IANA) to each TLV Type defines both the format and the semantic of the information carried in the TLV. Multiple TLVs may be encoded in the same SRH.

The highest-order bit of the TLV type (bit 0) specifies whether or not the TLV data of that type can change en route to the packet's final destination:

0: TLV data does not change en route

1: TLV data does change en route

All TLVs specify their alignment requirements using an xn+y format. The xn+y format is defined as per [RFC8200]. The SR Source nodes use the xn+y alignment requirements of TLVs and Padding TLVs when constructing an SRH.

The "Length" field of the TLV is used to skip the TLV while inspecting the SRH in case the node doesn't support or recognize the Type. The "Length" defines the TLV length in octets, not including the "Type" and "Length" fields.

The following TLVs are defined in this document:

Padding TLVs

HMAC TLV

Additional TLVs may be defined in the future.

2.1.1. Padding TLVs

There are two types of Padding TLVs, pad1 and padN, the following applies to both:

Padding TLVs are used for meeting the alignment requirement of the subsequent TLVs.

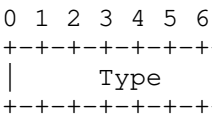
Padding TLVs are used to pad the SRH to a multiple of 8 octets.

Padding TLVs are ignored by a node processing the SRH TLV.

Multiple Padding TLVs MAY be used in one SRH

2.1.1.1. PAD1

Alignment requirement: none

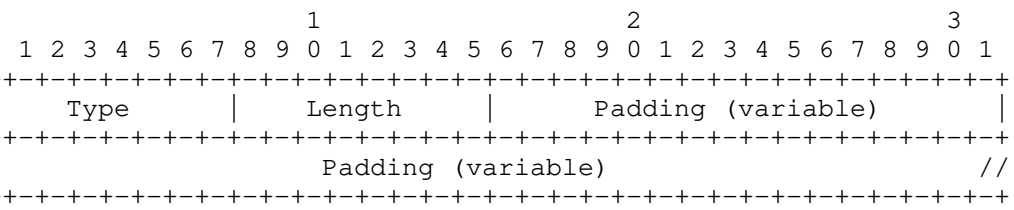


Type: to be assigned by IANA (Suggested value 0)

A single Pad1 TLV MUST be used when a single byte of padding is required. A Pad1 TLV MUST NOT be used if more than one consecutive byte of padding is required.

2.1.1.2. PADN

Alignment requirement: none



Type: to be assigned by IANA (suggested value 4).

Length: 0 to 5

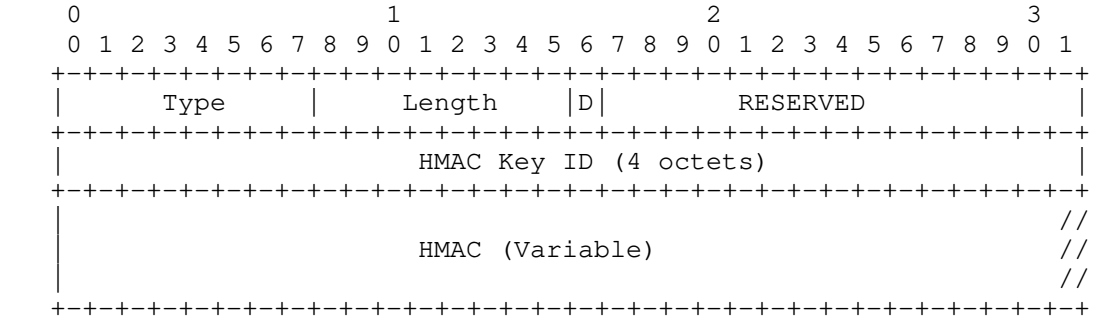
Padding: Length octets of padding. Padding bits have no semantic. They MUST be set to 0 on transmission and ignored on receipt.

The PadN TLV MUST be used when more than one byte of padding is required.

2.1.2. HMAC TLV

Alignment requirement: 8n

The keyed Hashed Message Authentication Code (HMAC) TLV is OPTIONAL and has the following format:



where:

- Type: to be assigned by IANA (suggested value 5).
- Length: The length of the variable length data in bytes.
- D: 1 bit. 1 indicates the Destination Address verification is disabled due to use of reduced segment list, Section 4.1.1.
- RESERVED: 15 bits. MUST be 0 on transmission.
- HMAC Key ID: A 4 octet opaque number which uniquely identifies the pre-shared key and algorithm used to generate the HMAC.
- HMAC: Keyed HMAC, in multiples of 8 octets, at most 32 octets.

The HMAC TLV is used to verify that the SRH applied to a packet was selected by an authorized party, and to ensure that the segment list is not modified after generation. This also allows for verification that the current segment (by virtue of being in the authorized segment list) is authorized for use. The SR Domain ensures the source node is permitted to use the source address in the packet via ingress filtering mechanisms as defined in BCP 84 [RFC3704], or other strategies as appropriate.

2.1.2.1. HMAC Generation and Verification

Local configuration determines when to check for an HMAC. This local configuration is outside the scope of this document. It may be based on the active segment at an SR Segment endpoint node, the result of an ACL that considers incoming interface, HMAC Key ID, or other packet fields.

An implementation that supports the generation and verification of the HMAC supports the following default behavior, as defined in the remainder of this section.

The HMAC verification begins by checking the current segment is equal to the destination address of the IPv6 header. The check is successful when either

- HMAC D bit is 1 and Segments Left is greater than Last Entry.
- HMAC Segments Left is less than or equal to Last Entry and destination address is equal to Segment List [Segments Left].

The HMAC field is the output of the HMAC computation as defined in [RFC2104], using:

- key: the pre-shared key identified by HMAC Key ID
- HMAC algorithm: identified by the HMAC Key ID
- Text: a concatenation of the following fields from the IPv6 header and the SRH, as it would be received at the node verifying the HMAC:
  - IPv6 header: source address (16 octets)
  - SRH: Last Entry (1 octet)
  - SRH: Flags (1 octet)
  - SRH: HMAC 16 bits following Length
  - SRH: HMAC Key ID (4 octets)
  - SRH: all addresses in the Segment List (variable octets)

The HMAC digest is truncated to 32 octets and placed in the HMAC field of the HMAC TLV.

For HMAC algorithms producing digests less than 32 octets, the digest is placed in the lowest order octets of the HMAC field. Subsequent octets MUST be set to zero such that the HMAC length is a multiple of 8 octets.

If HMAC verification is successful, processing proceeds as normal.

If HMAC verification fails, an ICMP error message (parameter problem, error code 0, pointing to the HMAC TLV) SHOULD be generated (but rate limited) and SHOULD be logged and the packet discarded.

2.1.2.2. HMAC Pre-Shared Key Algorithm

The HMAC Key ID field allows for the simultaneous existence of several hash algorithms (SHA-256, SHA3-256 ... or future ones) as well as pre-shared keys.

The HMAC Key ID field is opaque, i.e., it has neither syntax nor semantic except as an identifier of the right combination of pre-shared key and hash algorithm.

At the HMAC TLV generating and verification nodes, the Key ID uniquely identifies the pre-shared key and HMAC algorithm.

At the HMAC TLV generating node, the Text for the HMAC computation is set to the IPv6 header fields and SRH fields as they would appear at the verification node(s), not necessarily the same as the source node sending a packet with the HMAC TLV.

Pre-shared key roll-over is supported by having two key IDs in use while the HMAC TLV generating node and verifying node converge to a new key.

The HMAC TLV generating node may need to revoke an SRH for which it previously generated an HMAC. Revocation is achieved by allocating a new key and key ID, then rolling over the key ID associated with the SRH to be revoked. The HMAC TLV verifying node drops packets with the revoked SRH.

An implementation supporting HMAC can support multiple hash functions. An implementation supporting HMAC MUST implement SHA-2 [FIPS180-4] in its SHA-256 variant.

The selection of pre-shared key and algorithm, and their distribution is outside the scope of this document. Some options may include:

- in the configuration of the HMAC generating or verifying nodes, either by static configuration or any SDN oriented approach

- o dynamically using a trusted key distribution protocol such as [RFC6407]

While key management is outside the scope of this document, the recommendations of BCP 107 [RFC4107] should be considered when choosing the key management system.

3. SR Nodes

There are different types of nodes that may be involved in segment routing networks: source SR nodes originate packets with a segment in the destination address of the IPv6 header, transit nodes that forward packets destined to a remote segment, and SR segment endpoint nodes that process a local segment in the destination address of an IPv6 header.

3.1. Source SR Node

A Source SR Node is any node that originates an IPv6 packet with a segment (i.e. SRv6 SID) in the destination address of the IPv6 header. The packet leaving the source SR Node may or may not contain an SRH. This includes either:

A host originating an IPv6 packet.

An SR domain ingress router encapsulating a received packet in an outer IPv6 header, followed by an optional SRH.

The mechanism through which a segment in the destination address of the IPv6 header and the Segment List in the SRH, is derived is outside the scope of this document.

3.2. Transit Node

A transit node is any node forwarding an IPv6 packet where the destination address of that packet is not locally configured as a segment nor a local interface. A transit node is not required to be capable of processing a segment nor SRH.

3.3. SR Segment Endpoint Node

A SR segment endpoint node is any node receiving an IPv6 packet where the destination address of that packet is locally configured as a segment or local interface.

4. Packet Processing

This section describes SRv6 packet processing at the SR source, Transit and SR segment endpoint nodes.

4.1. Source SR Node

A Source node steers a packet into an SR Policy. If the SR Policy results in a segment list containing a single segment, and there is no need to add information to the SRH flag or to add TLV, the DA is set to the single segment list entry and the SRH MAY be omitted.

When needed, the SRH is created as follows:

Next Header and Hdr Ext Len fields are set as specified in [RFC8200].

Routing Type field is set as TBD (to be allocated by IANA, suggested value 4).

The DA of the packet is set with the value of the first segment.

The first element of the SRH Segment List is the ultimate segment. The second element is the penultimate segment, and so on.

The Segments Left field is set to n-1 where n is the number of elements in the SR Policy.

The Last Entry field is set to n-1 where n is the number of elements in the SR Policy.

TLVs (including HMAC) may be set according to their specification.

The packet is forwarded toward the packet's Destination Address (the first segment).

4.1.1. Reduced SRH

When a source does not require the entire SID list to be preserved in the SRH, a reduced SRH may be used.

A reduced SRH does not contain the first segment of the related SR Policy (the first segment is the one already in the DA of the IPv6 header), and the Last Entry field is set to n-2 where n is the number of elements in the SR Policy.

4.2. Transit Node

As specified in [RFC8200], the only node allowed to inspect the Routing Extension Header (and therefore the SRH), is the node corresponding to the DA of the packet. Any other transit node MUST NOT inspect the underneath routing header and MUST forward the packet toward the DA according to its IPv6 routing table.

When a SID is in the destination address of an IPv6 header of a packet, it's routed through an IPv6 network as an IPv6 address. SIDs, or the prefix(es) covering SIDs, and their reachability may be distributed by means outside the scope of this document. For example, [RFC5308] or [RFC5340] may be used to advertise a prefix covering the SIDs on a node.

4.3. SR Segment Endpoint Node

Without constraining the details of an implementation, the SR segment endpoint node creates Forwarding Information Base (FIB) entries for its local SIDs.

When an SRv6-capable node receives an IPv6 packet, it performs a longest-prefix-match lookup on the packets destination address. This lookup can return any of the following:

* A FIB entry that represents a locally instantiated SRv6 SID
* A FIB entry that represents a local interface, not locally instantiated as an SRv6 SID
* A FIB entry that represents a non-local route
* No Match

4.3.1. FIB Entry Is Locally Instantiated SRv6 SID

This document, and section, defines a single SRv6 SID. Future documents may define additional SRv6 SIDs. In which case, the entire content of this section will be defined in that document.

If the FIB entry represents a locally instantiated SRv6 SID, process the next header chain of the IPv6 header as defined in section 4 of [RFC8200]. Section 4.3.1.1 describes how to process an SRH, Section 4.3.1.2 describes how to process an upper layer header or no next header.

Processing this SID modifies the Segments Left and, if configured to process TLVs, it may modify the "variable length data" of TLV types that change en route. Therefore Segments Left is mutable and TLVs that change en route are mutable. The remainder of the SRH (Flags,

Tag, Segment List, and TLVs that do not change en route) are immutable while processing this SID.

4.3.1.1. SRH Processing

```
S01. When an SRH is processed {
S02.   If Segments Left is equal to zero {
S03.     Proceed to process the next header in the packet,
         whose type is identified by the Next Header field in
         the Routing header.
S04.   }
S05.   Else {
S06.     If local configuration requires TLV processing {
S07.       Perform TLV processing (see TLV Processing)
S08.     }
S09.     max_last_entry  =  ( Hdr Ext Len /  2 ) - 1
S10.     If  ((Last Entry > max_last_entry) or
S11.          (Segments Left is greater than (Last Entry+1)) {
S12.       Send an ICMP Parameter Problem, Code 0, message to
           the Source Address, pointing to the Segments Left
           field, and discard the packet.
S13.     }
S14.     Else {
S15.       Decrement Segments Left by 1.
S16.       Copy Segment List[Segments Left] from the SRH to the
           destination address of the IPv6 header.
S17.       If the IPv6 Hop Limit is less than or equal to 1 {
S18.         Send an ICMP Time Exceeded -- Hop Limit Exceeded in
             Transit message to the Source Address and discard
             the packet.
S19.       }
S20.       Else {
S21.         Decrement the Hop Limit by 1
S22.         Resubmit the packet to the IPv6 module for transmission
             to the new destination.
S23.       }
S24.     }
S25.   }
S26. }
```

4.3.1.1.1. TLV Processing

Local configuration determines how TLVs are to be processed when the Active Segment is a local SID defined in this document. The definition of local configuration is outside the scope of this document.

For illustration purpose only, two example local configurations that may be associated with a SID are provided below.

```
Example 1:
For any packet received from interface I2
  Skip TLV processing

Example 2:
For any packet received from interface I1
  If first TLV is HMAC {
    Process the HMAC TLV
  }
  Else {
    Discard the packet
  }
```

4.3.1.2. Upper-layer Header or No Next Header

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 SID defined in this document.

```
IF (Upper-layer Header is IPv4 or IPv6) and
    local configuration permits {
  Perform IPv6 decapsulation
  Resubmit the decapsulated packet to the IPv4 or IPv6 module
}
ELSE {
  Send an ICMP parameter problem message to the Source Address and
  discard the packet.  Error code (TBD by IANA) "SR Upper-layer
  Header Error", pointer set to the offset of the upper-layer
  header.
}
```

A unique error code allows an SR Source node to recognize an error in SID processing at an endpoint.

4.3.2. FIB Entry Is A Local Interface

If the FIB entry represents a local interface, not locally instantiated as an SRv6 SID, the SRH is processed as follows:

> If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing Header.
>
> If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

4.3.3. FIB Entry Is A Non-Local Route

Processing is not changed by this document.

4.3.4. FIB Entry Is A No Match

Processing is not changed by this document.

5. Intra SR Domain Deployment Model

The use of the SIDs exclusively within the SR Domain and solely for packets of the SR Domain is an important deployment model.

This enables the SR Domain to act as a single routing system.

This section covers:

- securing the SR Domain from external attempt to use its SIDs
- SR Domain as a single system with delegation between components
- handling packets of the SR Domain

5.1. Securing the SR Domain

Nodes outside the SR Domain are not trusted: they cannot directly use the SIDs of the domain. This is enforced by two levels of access control lists:

1. Any packet entering the SR Domain and destined to a SID within the SR Domain is dropped. This may be realized with the following logic. Other methods with equivalent outcome are considered compliant:
   * allocate all the SID's from a block S/s
   * configure each external interface of each edge node of the domain with an inbound infrastructure access list (IACL) which drops any incoming packet with a destination address in S/s
   * Failure to implement this method of ingress filtering exposes the SR Domain to source routing attacks as described and referenced in [RFC5095]
2. The distributed protection in #1 is complemented with per node protection, dropping packets to SIDs from source addresses outside the SR Domain. This may be realized with the following

logic. Other methods with equivalent outcome are considered compliant:

* assign all interface addresses from prefix A/a

* at node k, all SIDs local to k are assigned from prefix Sk/sk

* configure each internal interface of each SR node k in the SR Domain with an inbound IACL which drops any incoming packet with a destination address in Sk/sk if the source address is not in A/a.

5.2. SR Domain as A Single System with Delegation Among Components

All intra SR Domain packets are of the SR Domain. The IPv6 header is originated by a node of the SR Domain, and is destined to a node of the SR Domain.

All inter domain packets are encapsulated for the part of the packet journey that is within the SR Domain. The outer IPv6 header is originated by a node of the SR Domain, and is destined to a node of the SR Domain.

As a consequence, any packet within the SR Domain is of the SR Domain.

The SR Domain is a system in which the operator may want to distribute or delegate different operations of the outer most header to different nodes within the system.

An operator of an SR domain may choose to delegate SRH addition to a host node within the SR domain, and validation of the contents of any SRH to a more trusted router or switch attached to the host. Consider a top of rack switch (T) connected to host (H) via interface (I). H receives an SRH (SRH1) with a computed HMAC via some SDN method outside the scope of this document. H classifies traffic it sources and adds SRH1 to traffic requiring a specific SLA. T is configured with an IACL on I requiring verification of the SRH for any packet destined to the SID block of the SR Domain (S/s). T checks and verifies that SRH1 is valid, contains an HMAC TLV and verifies the HMAC.

An operator of the SR Domain may choose to have all segments in the SR Domain verify the HMAC. This mechanism would verify that the SRH segment list is not modified while traversing the SR Domain.

5.3. MTU Considerations

An SR Domain ingress edge node encapsulates packets traversing the SR Domain, and needs to consider the MTU of the SR Domain. Within the SR Domain, well known mitigation techniques are RECOMMENDED, such as deploying a greater MTU value within the SR Domain than at the ingress edges.

Encapsulation with an outer IPv6 header and SRH share the same MTU and fragmentation considerations as IPv6 tunnels described in [RFC2473]. Further investigation on the limitation of various tunneling methods (including IPv6 tunnels) are discussed in [I-D.ietf-intarea-tunnels] and SHOULD be considered by operators when considering MTU within the SR Domain.

5.4. ICMP Error Processing

ICMP error packets generated within the SR Domain are sent to source nodes within the SR Domain. The invoking packet in the ICMP error message may contain an SRH. Since the destination address of a packet with an SRH changes as each segment is processed, it may not be the destination used by the socket or application that generated the invoking packet.

For the source of an invoking packet to process the ICMP error message, the ultimate destination address of the IPv6 header may be required. The following logic is used to determine the destination address for use by protocol error handlers.

- Walk all extension headers of the invoking IPv6 packet to the routing extension header preceding the upper layer header.
  - If routing header is type TBD IANA (SRH)
    - The SID at Segment List[0] may be used as the destination address of the invoking packet.

ICMP errors are then processed by upper layer transports as defined in [RFC4443].

For IP packets encapsulated in an outer IPv6 header, ICMP error handling is as defined in [RFC2473].

5.5. Load Balancing and ECMP

For any inter domain packet, the SR Source node MUST impose a flow label computed based on the inner packet. The computation of the

flow label is as recommended in [RFC6438] for the sending Tunnel End Point.

For any intra domain packet, the SR Source node SHOULD impose a flow label computed as described in [RFC6437] to assist ECMP load balancing at transit nodes incapable of computing a 5-tuple beyond the SRH.

At any transit node within an SR domain, the flow label MUST be used as defined in [RFC6438] to calculate the ECMP hash toward the destination address. If flow label is not used, the transit node would likely hash all packets between a pair of SR Edge nodes to the same link.

At an SR segment endpoint node, the flow label MUST be used as defined in [RFC6438] to calculate any ECMP hash used to forward the processed packet to the next segment.

5.6. Other Deployments

Other deployment models and their implications on security, MTU, HMAC, ICMP error processing and interaction with other extension headers are outside the scope of this document.

6. Illustrations

This section provides illustrations of SRv6 packet processing at SR source, transit and SR segment endpoint nodes.

6.1. Abstract Representation of an SRH

For a node k, its IPv6 address is represented as Ak, its SRv6 SID is represented as Sk.

IPv6 headers are represented as the tuple of (source, destination). For example, a packet with source address A1 and destination address A2 is represented as (A1,A2). The payload of the packet is omitted.

An SR Policy is a list of segments. A list of segments is represented as <S1,S2,S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit.

(SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:

- Source Address is SA, Destination Addresses is DA, and next-header is SRH.
- SRH with SID list <S1, S2, S3> with SegmentsLeft = SL.

- Note the difference between the <> and () symbols. <S1, S2, S3> represents a SID list where the leftmost segment is the first segment. Whereas, (S3, S2, S1; SL) represents the same SID list but encoded in the SRH Segment List format where the leftmost segment is the last segment. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of detailed behavior, the (S3, S2, S1; SL) notation is more convenient.

At its SR Policy headend, the Segment List <S1,S2,S3> results in SRH (S3,S2,S1; SL=2) represented fully as:

```
Segments Left=2
Last Entry=2
Flags=0
Tag=0
Segment List[0]=S3
Segment List[1]=S2
Segment List[2]=S1
```

6.2. Example Topology

The following topology is used in examples below:

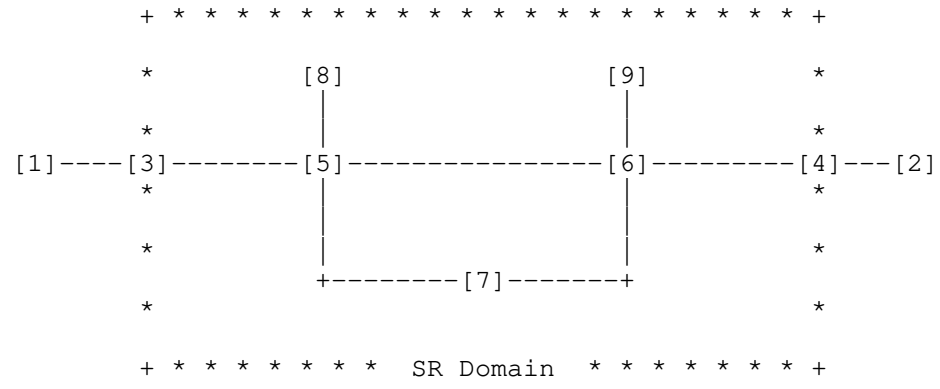


Figure 3

- 3 and 4 are SR Domain edge routers
- 5, 6, and 7 are all SR Domain routers
- 8 and 9 are hosts within the SR Domain
- 1 and 2 are hosts outside the SR Domain

- The SR domain implements ingress filtering as per Section 5.1 and no external packet can enter the domain with a destination address equal to a segment of the domain.

6.3. Source SR Node

6.3.1. Intra SR Domain Packet

When host 8 sends a packet to host 9 via an SR Policy <S7,A9> the packet is

P1: (A8,S7)(A9,S7; SL=1)

6.3.1.1. Reduced Variant

When host 8 sends a packet to host 9 via an SR Policy <S7,A9> and it wants to use a reduced SRH, the packet is

P2: (A8,S7)(A9; SL=1)

6.3.2. Inter SR Domain Packet - Transit

When host 1 sends a packet to host 2, the packet is

P3: (A1,A2)

The SR Domain ingress router 3 receives P3 and steers it to SR Domain egress router 4 via an SR Policy <S7, S4>. Router 3 encapsulates the received packet P3 in an outer header with an SRH. The packet is

P4: (A3, S7)(S4, S7; SL=1)(A1, A2)

If the SR Policy contains only one segment (the egress router 4), the ingress Router 3 encapsulates P3 into an outer header (A3, S4) without SRH. The packet is

P5: (A3, S4)(A1, A2)

6.3.2.1. Reduced Variant

The SR Domain ingress router 3 receives P3 and steers it to SR Domain egress router 4 via an SR Policy <S7, S4>. If router 3 wants to use a reduced SRH, Router 3 encapsulates the received packet P3 in an outer header with a reduced SRH. The packet is

P6: (A3, S7)(S4; SL=1)(A1, A2)

6.3.3. Inter SR Domain Packet - Internal to External

When host 8 sends a packet to host 1, the packet is encapsulated for the portion of its journey within the SR Domain. From 8 to 3 the packet is

P7: (A8,S3)(A8,A1)

In the opposite direction, the packet generated from 1 to 8 is

P8: (A1,A8)

At node 3 P8 is encapsulated for the portion of its journey within the SR domain, with the outer header destined to segment S8. Resulting in

P9: (A3,S8)(A1,A8)

At node 8 the outer IPv6 header is removed by S8 processing, then processed again when received by A8.

6.4. Transit Node

Nodes 5 acts as transit nodes for packet P1, and sends packet

P1: (A8,S7)(A9,S7;SL=1)

on the interface toward node 7.

6.5. SR Segment Endpoint Node

Node 7 receives packet P1 and, using the logic in Section 4.3.1, sends packet

P7: (A8,A9)(A9,S7; SL=0)

on the interface toward router 6.

6.6. Delegation of Function with HMAC Verification

This section describes how a function may be delegated within the SR Domain. In the following sections consider a host 8 connected to a top of rack 5.

6.6.1. SID List Verification

An operator may prefer to apply the SRH at source 8, while 5 verifies the SID list is valid.

For illustration purpose, an SDN controller provides 8 an SRH terminating at node 9, with segment list <S5,S7,S6,A9>, and HMAC TLV computed for the SRH. The HMAC key ID and key associated with the HMAC TLV is shared with 5. Node 8 does not know the key. Node 5 is configured with an IACL applied to the interface connected to 8, requiring HMAC verification for any packet destined to S/s.

Node 8 originates packets with the received SRH including HMAC TLV.

P15:(A8,S5)(A9,S6,S7,S5;SL=3;HMAC)

Node 5 receives and verifies the HMAC for the SRH, then forwards the packet to the next segment

P16:(A8,S7)(A9,S6,S7,S5;SL=2;HMAC)

Node 6 receives

P17:(A8,S6)(A9,S6,S7,S5;SL=1;HMAC)

Node 9 receives

P18:(A8,A9)(A9,S6,S7,S5;SL=0;HMAC)

This use of an HMAC is particularly valuable within an enterprise based SR Domain [SRN].

7. Security Considerations

This section reviews security considerations related to the SRH, given the SRH processing and deployment models discussed in this document.

As described in Section 5, it is necessary to filter packets ingress to the SR Domain, destined to SIDs within the SR Domain (i.e., bearing a SID in the destination address). This ingress filtering is via an IACL at SR Domain ingress border nodes. Additional protection is applied via an IACL at each SR Segment Endpoint node, filtering packets not from within the SR Domain, destined to SIDs in the SR Domain. ACLs are easily supported for small numbers of prefixes, making summarization important, and when the prefixes requiring filtering is kept to a seldom changing set.

Additionally, ingress filtering of IPv6 source addresses as recommended in BCP38 [RFC2827] SHOULD be used.

7.1. Source Routing Attacks

An SR domain implements distributed and per node protection as described in section 5.1. Additionally, domains deny traffic with spoofed addresses by implementing the recommendations in BCP 84 [RFC3704].

Full implementation of the recommended protection blocks the attacks documented in [RFC5095] from outside the SR domain, including bypassing filtering devices, reaching otherwise unreachable Internet systems, network topology discovery, bandwidth exhaustion, and defeating anycast.

Failure to implement distributed and per node protection allows attackers to bypass filtering devices and exposes the SR Domain to these attacks.

Compromised nodes within the SR Domain may mount the attacks listed above along with other known attacks on IP networks (e.g. DOS/DDOS, topology discovery, man-in-the-middle, traffic interception/ siphoning).

7.2. Service Theft

Service theft is defined as the use of a service offered by the SR Domain by a node not authorized to use the service.

Service theft is not a concern within the SR Domain as all SR Source nodes and SR segment endpoint nodes within the domain are able to utilize the services of the Domain. If a node outside the SR Domain learns of segments or a topological service within the SR domain, IACL filtering denies access to those segments.

7.3. Topology Disclosure

The SRH is unencrypted and may contain SIDs of some intermediate SR-nodes in the path towards the destination within the SR Domain. If packets can be snooped within the SR Domain, the SRH may reveal topology, traffic flows, and service usage.

This is applicable within an SR Domain, but the disclosure is less relevant as an attacker has other means of learning topology, flows, and service usage.

7.4. ICMP Generation

The generation of ICMPv6 error messages may be used to attempt denial-of-service attacks by sending an error-causing destination address or SRH in back-to-back packets. An implementation that correctly follows Section 2.4 of [RFC4443] would be protected by the ICMPv6 rate-limiting mechanism.

7.5. Applicability of AH

The SR Domain is a trusted domain, as defined in [RFC8402] Section 2 and Section 8.2. The SR Source is trusted to add an SRH (optionally verified as having been generated by a trusted source via the HMAC TLV in this document), and segments advertised within the domain are trusted to be accurate and advertised by trusted sources via a secure control plane. As such the SR Domain does not rely on the Authentication Header (AH) as defined in [RFC4302] to secure the SRH.

The use of SRH with AH by an SR source node, and processing at a SR segment endpoint node is not defined in this document. Future documents may define use of SRH with AH and its processing.

8. IANA Considerations

This document makes the following registrations in the Internet Protocol Version 6 (IPv6) Parameters "Routing Type" registry maintained by IANA:

Suggested Value	Description	Reference
4	Segment Routing Header (SRH)	This document

This document makes the following registrations in "Type 4 - Parameter Problem" message of the "Internet Control Message Protocol version 6 (ICMPv6) Parameters" registry maintained by IANA:

CODE	NAME/DESCRIPTION
TBD IANA	SR Upper-layer Header Error

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the SRH, in accordance with BCP 26, [RFC8126].

The following terms are used here with the meanings defined in BCP 26: "namespace", "assigned value", "registration".

The following policies are used here with the meanings defined in BCP 26 [RFC8126]: "IETF Review".

8.1. Segment Routing Header Flags Registry

This document requests the creation of a new IANA managed registry to identify SRH Flags Bits. The registration procedure is "IETF Review". Suggested registry name is "Segment Routing Header Flags". Flags is 8 bits.

8.2. Segment Routing Header TLVs Registry

This document requests the creation of a new IANA managed registry to identify SRH TLVs. The registration procedure is "IETF Review". Suggested registry name is "Segment Routing Header TLVs". A TLV is identified through an unsigned 8 bit codepoint value, with assigned values 0-127 for TLVs that do not change en route, and 128-255 for TLVs that may change en route. The following codepoints are defined in this document:

Assigned Value	Description	Reference
0	Pad1 TLV	This document
1	Reserved	This document
2	Reserved	This document
3	Reserved	This document
4	PadN TLV	This document
5	HMAC TLV	This document
6	Reserved	This document
124-126	Experimentation and Test	This document
127	Reserved	This document
252-254	Experimentation and Test	This document
255	Reserved	This document

Values 1,2,3,6 were defined in draft versions of this specification and are Reserved for backwards compatibility with early implementations and should not be reassigned. Values 127 and 255 are Reserved to allow for expansion of the Type field in future specifications if needed.

9. Implementation Status

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

See [I-D.matsushima-spring-srv6-deployment-status] for updated deployment and interoperability reports.

9.1. Linux

Name: Linux Kernel v4.14

Status: Production

Implementation: adds SRH, performs END processing, supports HMAC TLV

Details: https://irtf.org/anrw/2017/anrw17-final3.pdf and [I-D.filsfils-spring-srv6-interop]

9.2. Cisco Systems

Name: IOS XR and IOS XE

Status: Production (IOS XR), Pre-production (IOS XE)

Implementation: adds SRH, performs END processing, no TLV processing

Details: [I-D.filsfils-spring-srv6-interop]

9.3. FD.io

Name: VPP/Segment Routing for IPv6

Status: Production

Implementation: adds SRH, performs END processing, no TLV processing

Details: https://wiki.fd.io/view/VPP/Segment_Routing_for_IPv6 and [I-D.filsfils-spring-srv6-interop]

9.4. Barefoot

Name: Barefoot Networks Tofino NPU

Status: Prototype

Implementation: performs END processing, no TLV processing

Details: [I-D.filsfils-spring-srv6-interop]

9.5. Juniper

Name: Juniper Networks Trio and vTrio NPU's

Status: Prototype & Experimental

Implementation: SRH insertion mode, Process SID where SID is an interface address, no TLV processing

9.6. Huawei

Name: Huawei Systems VRP Platform

Status: Production

Implementation: adds SRH, performs END processing, no TLV processing

10. Contributors

Kamran Raza, Zafar Ali, Brian Field, Daniel Bernier, Ida Leung, Jen Linkova, Ebben Aries, Tomoya Kosugi, Eric Vyncke, David Lebrun, Dirk Steinberg, Robert Raszuk, Dave Barach, John Brzozowski, Pierre Francois, Nagendra Kumar, Mark Townsley, Christian Martin, Roberta Maglione, James Connolly, Aloys Augustin contributed to the content of this document.

11. Acknowledgements

The authors would like to thank Ole Troan, Bob Hinden, Ron Bonica, Fred Baker, Brian Carpenter, Alexandru Petrescu, Punit Kumar Jaiswal, David Lebrun, Benjamin Kaduk, Frank Xialiang, Mirja Kuhlewind, Roman Danyliw, Joe Touch, and Magnus Westerlund for their comments to this document.

12. References

12.1. Normative References

[FIPS180-4] National Institute of Standards and Technology, "FIPS 180-4 Secure Hash Standard (SHS)", March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

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

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <https://www.rfc-editor.org/info/rfc2827>.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <https://www.rfc-editor.org/info/rfc3704>.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <https://www.rfc-editor.org/info/rfc4107>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <https://www.rfc-editor.org/info/rfc5095>.

[RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <https://www.rfc-editor.org/info/rfc6407>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <https://www.rfc-editor.org/info/rfc6437>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

12.2. Informative References

[I-D.filsfils-spring-srv6-interop]
Filsfils, C., Clad, F., Camarillo, P., Abdelsalam, A., Salsano, S., Bonaventure, O., Horn, J., and J. Liste, "SRv6 interoperability report", draft-filsfils-spring-srv6-interop-02 (work in progress), March 2019.

[I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.

[I-D.matsushima-spring-srv6-deployment-status]
Matsushima, S., Filsfils, C., Ali, Z., and Z. Li, "SRv6 Implementation and Deployment Status", draft-matsushima-spring-srv6-deployment-status-02 (work in progress), October 2019.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <https://www.rfc-editor.org/info/rfc5308>.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <https://www.rfc-editor.org/info/rfc5340>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[SRN] Lebrun, D., Jadin, M., Clad, F., Filsfils, C., and O. Bonaventure, "Software Resolved Networks: Rethinking Enterprise Networks with IPv6 Segment Routing", 2018, <https://inl.info.ucl.ac.be/system/files/sosr18-final15-embedfonts.pdf>.

Authors' Addresses

Clarence Filsfils (editor)
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Darren Dukes (editor)
Cisco Systems, Inc.
Ottawa
CA

Email: ddukes@cisco.com

Stefano Previdi
Huawei
Italy

Email: stefano@previdi.net

John Leddy
Individual
US

Email: john@leddy.net

Satoru Matsushima
Softbank

Email: satoru.matsushima@g.softbank.co.jp

Daniel Voyer
Bell Canada

Email: daniel.voyer@bell.ca

ippm,6man
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2019

S. Bhandari
F. Brockners
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JMPC
T. Mizrahi
Huawei Network.IO Innovation Lab
A. Kfir
B. Gafni
Mellanox Technologies, Inc.
P. Lapukhov
Facebook
M. Spiegel
Barefoot Networks
S. Krishnan
Kaloom

October 20, 2018

In-situ OAM IPv6 Options
draft-ioametal-ippm-6man-ioam-ipv6-options-01

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document outlines how IOAM data fields are encapsulated in IPv6.

This Internet-Draft will expire on April 23, 2019.

Table of Contents

1. Introduction

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document outlines how IOAM data fields are encapsulated in the IPv6 [RFC8200].

2. Conventions

2.1. Requirements Language

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

2.2. Abbreviations

Abbreviations used in this document:

E2E: Edge-to-Edge

IOAM: In-situ Operations, Administration, and Maintenance

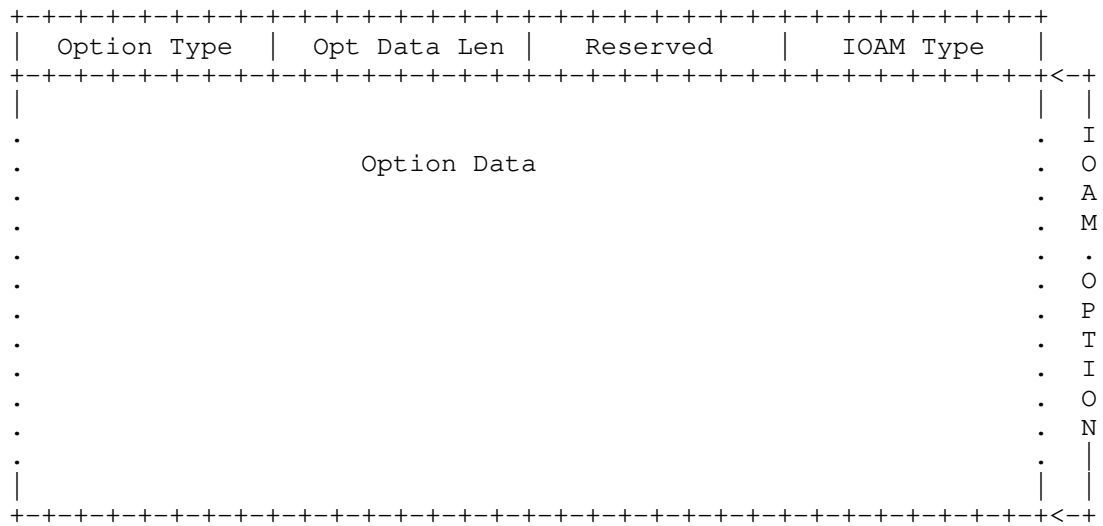
OAM: Operations, Administration, and Maintenance

POT: Proof of Transit

3. In-situ OAM Metadata Transport in IPv6

IOAM data is carried in IPv6 packets as Hop-by-Hop or Destination options. One IPv6 Destination Options and Hop-by-Hop Options Type codepoint is assigned for IOAM. Multiple options with the same Option Type MAY appear in the same Hop-by-Hop Options or Destination Options header, with varying content. This mechanism of in-situ OAM in IPv6 is used to enhance diagnostics of IPv6 networks. It complements other mechanisms proposed to enhance diagnostics of IPv6 networks, such as the IPv6 Performance and Diagnostic Metrics Destination Option described in [RFC8250].

IPv6 Hop-by-Hop and Destination Option format for carrying in-situ OAM data fields:



Option Type: 8-bit identifier of the type of option.

Opt Data Len: 8-bit unsigned integer. Length of the Reserved and Option Data field of this option, in octets.

Reserved: 8-bit field MUST be set to zero upon transmission and ignored upon reception.

IOAM Type: 8-bit field as defined in section 7.2 in [I-D.ietf-ippm-ioam-data].

Option Data: Variable-length field. Option-Type-specific data.

In-situ OAM Options are inserted as Option data as follows:

1. Pre-allocated Tracing Option: The in-situ OAM Preallocated Tracing option defined in [I-D.ietf-ippm-ioam-data] is represented as a IPv6 option in hop by hop extension header:

   Option Type: 001xxxxx 8-bit identifier of the IOAM type of option. xxxxx=TBD.

   IOAM Type: IOAM Pre-allocated Trace Option Type.

2. Incremental Tracing Option: The in-situ OAM Incremental Tracing option defined in [I-D.ietf-ippm-ioam-data] is represented as a IPv6 option in hop by hop extension header:

   Option Type: 001xxxxx 8-bit identifier of the IOAM type of option. xxxxx=TBD.

   IOAM Type: IOAM Incremental Trace Option Type.

3. Proof of Transit Option: The in-situ OAM POT option defined in [I-D.ietf-ippm-ioam-data] is represented as a IPv6 option in hop by hop extension header:

   Option Type: 001xxxxx 8-bit identifier of the IOAM type of option. xxxxx=TBD.

   IOAM Type: IOAM POT Option Type.

4. Edge to Edge Option: The in-situ OAM E2E option defined in [I-D.ietf-ippm-ioam-data] is represented as a IPv6 option in IPv6 option in destination options extension header:

   Option Type: 000xxxxx 8-bit identifier of the IOAM type of option. xxxxx=TBD.

IOAM Type: IOAM E2E Option Type.

All the in-situ OAM IPv6 options defined here have alignment requirements. Specifically, they all require 4n alignment. This ensures that 4 octet fields specified in [I-D.ietf-ippm-ioam-data] such as transit delay are aligned at a multiple-of-4 offset from the start of the Hop-by-Hop Options header. In addition, to maintain IPv6 extension header 8-octet alignment and avoid the need to add or remove padding at every hop, the Trace-Type for Incremental Tracing Option in IPv6 MUST be selected such that the IOAM node data length is a multiple of 8-octets.

4. Security Considerations

This document describes the encapsulation of IOAM data fields in IPv6. Security considerations of the specific IOAM data fields for each case (i.e., Trace, Proof of Transit, and E2E) are described in defined in [I-D.ietf-ippm-ioam-data].

As this document describes new options for IPv6 , these are similar to the security considerations of [RFC8200] and the new weakness documented in [RFC8250].

5. IANA Considerations

This draft requests the following IPv6 Option Type assignments from the Destination Options and Hop-by-Hop Options sub-registry of Internet Protocol Version 6 (IPv6) Parameters.

http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2

Hex Value	Binary Value act	chg	rest	Description	Reference
TBD_1_0	00	0	TBD_1	IOAM	[This draft]
TBD_1_1	00	1	TBD_1	IOAM	[This draft]

6. Acknowledgements

The authors would like to thank Tom Herbert, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Stefano Previdi, Hemant Singh, Erik Nordmark, LJ Wobker, and Andrew Yourtchenko for the comments and advice. For the IPv6 encapsulation, this document leverages concepts described in [I-D.kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

7. References

7.1. Normative References

[I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., and d. daniel.bernier@bell.ca, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-01 (work in progress), October 2017.

[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>.

[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>.

7.2. Informative References

[I-D.kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop Option Extension", draft-kitamura-ipv6-record-route-00 (work in progress), November 2000.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8250] Elkins, N., Hamilton, R., and M. Ackermann, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", RFC 8250, DOI 10.17487/RFC8250, September 2017, <https://www.rfc-editor.org/info/rfc8250>.

Authors' Addresses

Shwetha Bhandari
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Email: shwethab@cisco.com

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany

Email: fbrockne@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
7200-11 Kit Creek Road
Research Triangle Park, NC 27709
United States

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

Stephen Youell
JP Morgan Chase
25 Bank Street
London E14 5JP
United Kingdom

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Huawei Network.IO Innovation Lab
Israel

Email: tal.mizrahi.phd@gmail.com

Aviv Kfir
Mellanox Technologies, Inc.
350 Oakmead Parkway, Suite 100
Sunnyvale, CA 94085
U.S.A.

Email: avivk@mellanox.com

Barak Gafni
Mellanox Technologies, Inc.
350 Oakmead Parkway, Suite 100
Sunnyvale, CA 94085
U.S.A.

Email: gbarak@mellanox.com

Petr Lapukhov
Facebook
1 Hacker Way
Menlo Park, CA 94025
US

Email: petr@fb.com

Mickey Spiegel
Barefoot Networks
4750 Patrick Henry Drive
Santa Clara, CA 95054
US

Email: mspiegel@barefootnetworks.com

Suresh Krishnan
Kaloom

Email: suresh@kaloom.com

6man
Internet-Draft
Intended status: Standards Track
Expires: April 15, 2019

J. Leddy
Unaffiliated
R. Bonica
Juniper Networks
I. Lubashev
Akamai Technologies

October 12, 2018

IPv6 Packet Truncation
draft-leddy-6man-truncate-05

Abstract

This document defines IPv6 packet truncation procedures. These procedures make Path MTU Discovery (PMTUD) more reliable. Upper-layer protocols can leverage these procedures in order to take advantage of large MTUs.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 15, 2019.

Copyright Notice

Table of Contents

1. Introduction

An Internet path connects a source node to a destination node. A path can contain links and routers.

Each link is constrained by the number of bytes that it can convey in a single IP packet. This constraint is called the link Maximum Transmission Unit (MTU). IPv6 [RFC8200] requires every link to have an MTU of 1280 bytes or greater. This value is called IPv6 minimum link MTU.

Likewise, each Internet path is constrained by the number of bytes that it can convey in a single IP packet. This constraint is called the Path MTU (PMTU). For any given path, the PMTU is equal to the smallest of its link MTUs.

IPv6 allows fragmentation at the source node only. If an IPv6 source node sends a packet whose length exceeds the PMTU, an intermediate node will discard the packet. In order to prevent this, IPv6 nodes can either:

- Refrain from sending packets whose length exceeds the IPv6 minimum link MTU.

- Maintain a running estimate of the PMTU and refrain from sending packets whose length exceeds that estimate.

In order to maintain a running estimate of the PMTU, IPv6 nodes can execute Path MTU Discovery (PMTUD) [RFC8201] procedures. In these procedures, the source node produces an initial PMTU estimate. This initial estimate equals the MTU of the first link along the path to the destination. It can be greater than the actual PMTU.

Having produced an initial PMTU estimate, the source node sends packets to the destination node. If one of these packets is larger than the actual PMTU, an intermediate node will not be able to forward the packet through the next link along the path. Therefore, the intermediate node discards the packet and sends an Internet Control Message Protocol (ICMP) [RFC4443] Packet Too Big (PTB) message to the source node. The ICMP PTB message indicates the MTU of the link through which the packet could not be forwarded. The source node uses this information to refine its PMTU estimate.

PMTUD relies on the network to deliver ICMP PTB messages from the intermediate node to the source node. If the network cannot deliver these messages, a persistent black hole can develop. In this scenario, the source node sends a packet whose length exceeds the PMTU. An intermediate node discards the packet and sends an ICMP PTB message to the source. However, the network cannot deliver the ICMP PTB message to the source. Therefore, the source node does not update its PMTU estimate and it continues to send packets whose length exceeds the PMTU. The intermediate node discards these packets and sends more ICMP PTB messages to the source. These ICMP PTB messages are lost, exactly as previous ICMP PTB messages were lost.

In some operational scenarios (Section 3), networks cannot deliver ICMP PTB messages from an intermediate node to the source node. Therefore, enhanced procedures are required.

This document defines IPv6 packet truncation procedures. When an IPv6 source node originates a packet, it executes the following procedure:

- Mark the packet as being eligible for truncation.

- Forward the packet towards its destination.

If an intermediate node cannot forward the packet because of an MTU issue, it executes the following procedure:

- Detect that the packet is eligible for truncation.
- Send an ICMP PTB message to the source node, with the MTU field indicating the MTU of the link through which the packet could not be forwarded.
- Truncate the packet.
- Mark the packet as being truncated.
- Update the packet's upper-layer checksum (if possible).
- Forward the packet towards its destination.

When the destination node receives the packet, it executes the following procedure:

- Detect that the packet has been truncated.
- Send an ICMP PTB message to the source node, with the MTU field indicating the length of the truncated packet.
- Discard the packet.

Both ICMP PTB messages, mentioned above, contain MTU information that the source node can use to refine its PMTU estimate.

The procedures described herein prevent incomplete (i.e., truncated) data from being delivered to upper-layer protocols. While IPv6 packet truncation may facilitate new upper-layer procedures, upper-layer procedures are beyond the scope of this document.

The procedures described herein make PMTUD more reliable by increasing the probability that the source node will receive ICMP PTB feedback from a downstream device. Even when the network cannot deliver ICMP PTB messages from an intermediate router to a source node, it may be able to deliver an ICMP PTB messages from the destination node to the source node.

However, the procedures described herein do not make PMTUD one hundred per cent reliable. In some operational scenarios, the network cannot deliver any ICMP messages to the source node, regardless of their origin.

2. Requirements Language

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

3. Operational Considerations

The packet truncation procedures described herein make PMTUD more resilient when:

- The network can deliver ICMP messages from the destination node to the source node.
- The network cannot deliver ICMP messages from an intermediate node to the source node.

The following are common operational scenarios in which packet truncation procedures can make PMTUD more resilient:

- The destination node has a viable route to the source node, but the intermediate node does not.
- The source node is protected by a firewall that administratively blocks all packets except for those from specified subnetworks. The destination node resides in one of the specified subnetworks, but the intermediate node does not.
- The source address of the original packet (i.e., the packet that elicited the ICMP message) was an anycast address. Therefore, the destination address of the ICMP message is the same anycast address. In this case, an ICMP message from the destination node is likely to be delivered to the correct anycast instance. By contrast, an ICMP message from an intermediate node is less likely to be delivered to the correct anycast instance.

Packet truncation procedures do not make PMTUD more resilient when the network cannot reliably deliver any ICMP messages to the source node. The following are operational scenarios where the network cannot reliably deliver any ICMP PTB messages to the source node:

- The source node is protected by a firewall that administratively blocks all ICMP messages.

- The source node is an anycast instance served by a load-balancer as defined in [RFC7690]. The load-balancer does not implement the mitigations defined in [RFC7690].

4. IPv6 Destination Options

This document defines the following IPv6 Destination options:

4.1. The Truncation Eligible Option

The Truncation Eligible option indicates that the packet is eligible for truncation. It also indicates that the packet has not been truncated.

The Truncation Eligible option contains the following fields:

- Option Type - Truncation Eligible option. Value TBD by IANA. See Notes below.

- Opt Data Len - Length of Option Data, measured in bytes. MUST be equal to 0.

IPv6 packets that include the Fragment header MUST NOT include the Truncation Eligible option.

IPv6 packets whose length is less than the IPv6 minimum link MTU SHOULD NOT include the Truncation Eligible option.

The IPv6 Hop-by-hop Options header SHOULD NOT include the Truncation Eligible option.

The IPv6 Destination Options header:

- MAY include a single instance of the Truncation Eligible option.

- SHOULD NOT include multiple instances of the Truncation Eligible option.

- MUST NOT include both the Truncation Eligible option and the Truncated Packet option (Section 4.2).

NOTE 1: According to [RFC8200], the highest-order two bits of the Option Type (i.e., the "act" bits) specify the action taken by a processing node that does not recognize Option Type. The required action is skip over this option and continue processing the header. Therefore, IANA is requested to assign this Option Type with "act" bits "00".

NOTE 2: According to [RFC8200], the third-highest-order bit (i.e., the "chg" bit) of the Option Type specifies whether Option Data can change on route to the packet's destination. Because this option contains no Option Data, IANA can assign this Option Type without regard to the "chg" bit.

4.2. The Truncated Packet Option

The Truncated Packet option indicates that the packet has been truncated and is eligible for further truncation.

The Truncated Packet option contains the following fields:

- Option Type - Truncated Packet option. Value TBD by IANA. See Notes below.
- Opt Data Len - Length of Option Data, measured in bytes. MUST be equal to 0.

IPv6 packets that include the Fragment header MUST NOT include the Truncated Packet option.

IPv6 packets whose length is less than the IPv6 minimum link MTU MUST NOT include the Truncated Packet option.

The IPv6 Hop-by-hop Options header SHOULD NOT include the Truncated Packet option.

The IPv6 Destination Options:

- MAY include a single instance of the Truncated Packet option.
- SHOULD NOT include multiple instances of the Truncated Packet option.
- MUST NOT include both the Truncated Packet option and the Truncation Eligible option.

NOTE 1: According to [RFC8200], the highest-order two bits of the Option Type (i.e., the "act" bits) specify the action taken by a processing node that does not recognize Option Type. The required action is to discard the packet and send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type. Therefore, IANA is requested to assign this Option Type with "act" bits "10".

NOTE 2: According to [RFC8200], the third-highest-order bit (i.e., the "chg" bit) of the Option Type specifies whether Option Data of

that option can change on route to the packet's destination. Because this option contains no Option Data, IANA can assign this Option Type without regard to the "chg" bit.

5. Reference Topology

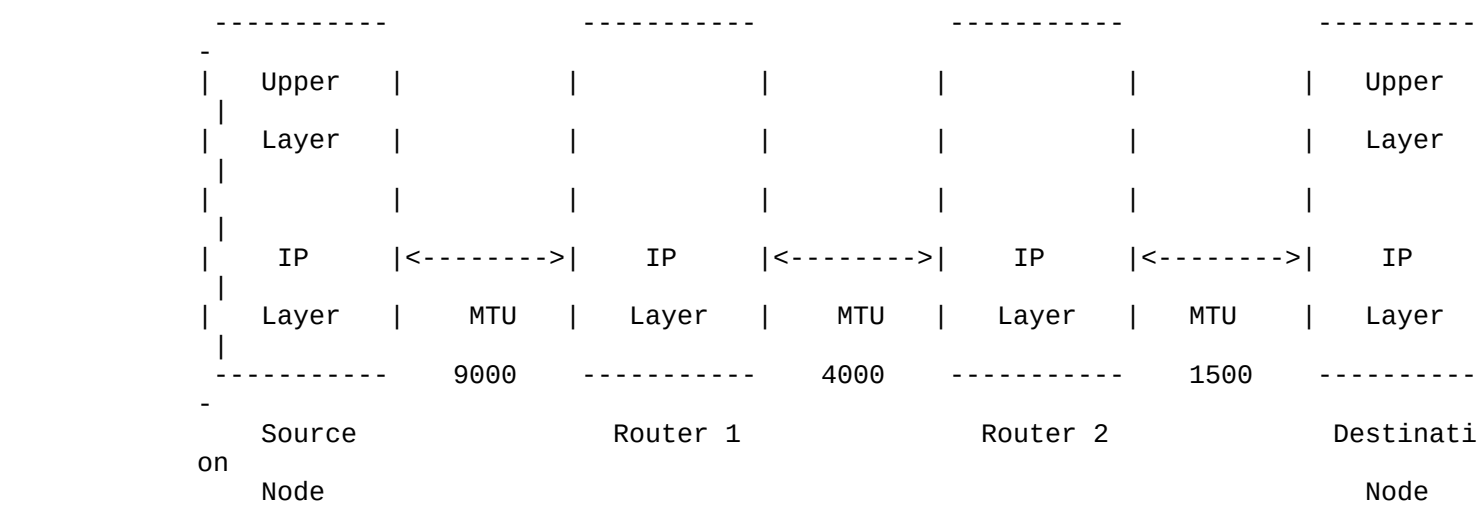


Figure 1: Reference Topology

Figure 1 depicts a network that contains a Source Node, intermediate nodes (i.e., Router 1, Router 2), and a Destination Node. The link that connects the Source Node to Router 1 has an MTU of 9000 bytes. The link that connects Router 1 to Router 2 has an MTU of 4000 bytes, and the link that connects Router 2 to the Destination Node has an MTU of 1500 bytes. The PMTU between the Source Node and the Destination Node is 1500 bytes.

This topology is used in examples throughout the document.

6. Truncation Procedures

In the Reference Topology (Figure 1), the Source Node produces an initial estimate of the PMTU between itself and the Destination Node. This initial estimate equals the MTU of the first link on the path to the Destination Node (e.g., 9000 bytes).

The Source Node refrains from sending packets whose length exceeds the above-mentioned estimate. However, the above-mentioned estimate is significantly larger than the actual PMTU (1500 bytes). Therefore, the Source Node may send packets whose length exceeds the actual PMTU.

At some time in the future, an upper-layer protocol on the Source Node causes the IP layer to emit a packet. The packet contains a Destination Options header and the Destination Options header contains a Truncation Eligible option. The total packet length, including all headers and the payload, is 1350 bytes. Because the total packet length is less than the actual PMTU, this packet can be

delivered to the Destination Node without encountering any MTU issues.

The IP layer on the Source Node forwards the packet to the Router 1, Router 1 forwards the packet to Router 2, and the Router 2 forwards the packet to the Destination Node. The IP layer on the Destination Node examines the Destination Options header and finds the Truncation Eligible option. The Truncation Eligible option requires no action by the Destination Node. Therefore, the Destination Node processes the next header and delivers the packet to an upper-layer protocol.

Subsequently, the same upper-layer protocol on the Source Node causes the IP layer to emit another packet. This packet is identical to the first, except that the total packet length is 2000 bytes. Because the packet length is greater than the actual PMTU, this packet cannot be delivered without encountering an MTU issue.

The IP layer on the source node forwards the packet to Router 1. Router 1 forwards the packet to Router 2, but the Router 2 cannot forward the packet because its length exceeds the MTU of the next link in the path (i.e., 1500 bytes). Because an MTU issue has been encountered, Router 2 examines the Destination Options header, searching for either a Truncation Eligible option or a Truncated Packet option. (Normally, the Router 2 would ignore the Destination Options header).

Because Router 2 finds one of the above-mentioned options, it:

- Sends an ICMP PTB message to the Source Node. The ICMP PTB message contains an MTU field indicating the MTU of the next link in the path (i.e. 1500 bytes).

- Truncates the packet, so that its total length equals the MTU of the next link in the path.

- Updates the IPv6 Payload Length.

- Overwrites all instances of the Truncation Eligible option with a Truncated Packet option.

- Updates the upper-layer checksum (if possible)

- Forwards the packet to the Destination Node.

The IP layer on the Destination Node receives the packet and examines the Destination Options header. Because it finds the Truncated Packet option, it discards the packet and sends an ICMP PTB message

to the Source Node. The MTU field in the ICMP PTB message represents the length of the received packet.

When the Source Node receives the ICMP PTB message, it updates its PMTU estimate, as per [RFC8201].

7. Additional Truncation Considerations

A packet can be truncated multiple times. In the Reference Topology (Figure 1), assume that the Source Node sends a 5000 byte packet to the Destination Node. Using the procedures described in Section 6, Router 1 truncates this packet to 4000 bytes and Router 2 truncates it again, to 1500 bytes.

A truncated packet MUST contain the basic IPv6 header, all extension headers and the first upper-layer header. When an intermediate node cannot forward a packet due to MTU issues, and the total length of the basic IPv6 header, all extension headers, and first upper-layer header exceeds the MTU of the next link in the path, the intermediate node MUST discard the packet and send and ICMP PTB message to the source node. It MUST NOT truncate the packet.

A truncated packet MUST NOT include the Fragment header. When an intermediate node cannot forward a packet due to MTU issues, and the packet contains a Fragment header, the intermediate node MUST discard the packet and send and ICMP PTB message to the source node. It MUST NOT truncate the packet.

A truncated packet must have a total length that is greater than or equal to the IPv6 minimum link MTU.

8. Backwards Compatibility

Section 6 of this document assumes that all nodes recognize the Truncation Eligible and Truncated Packet options. This section explores backwards compatibility issues, where one or more nodes do not recognize the above-mentioned options.

An intermediate node that does not recognize the above-mentioned options behaves exactly as described in [RFC8200]. When it receives a packet that does not cause an MTU issue, it processes the packet. When it receives a packet that causes an MTU issue, it discards the packet and sends an ICMP PTB message to the source node. In neither case does the intermediate node examine the Destination Options header or truncate the packet.

A destination node that does not recognize the Truncation Eligible option also behaves exactly as described in [RFC8200]. When it

receives a packet that contains the Truncation Eligible option, its behavior is determined by the highest-order two bits of the Option Type (i.e., the "act" bits). Because the "act" bits are equal to "00", the destination node skips over the option and continues to process the packet. This is exactly what the destination node would have done if it had recognized the Truncation Eligible option.

A destination node that does not recognize the Truncated Packet option also behaves exactly as described in [RFC8200]. When it receives a packet that contains the Truncated Packet option, its behavior is determined by the highest-order two bits of the Option Type (i.e., the "act" bits). Because the "act" bits are equal to "10", the destination node discards the packet and sends an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the Truncated Packet option. The destination node does not emit an ICMP PTB message.

The source node takes appropriate action when it receives the ICMP Parameter Problem message.

9. Checksum Considerations

When an intermediate node truncates a packet, it SHOULD update the upper-layer checksum, if possible. This is desirable because it increases the probability that the truncated packet will be delivered to the destination node.

Middleboxes residing downstream of the intermediate node may attempt to validate the upper-layer checksum. If validation fails, they may discard the packet without sending an ICMP message.

10. Invalid Packet Types

The following packet types are invalid:

- Packets that contain the Fragment header and the Truncation Eligible option.
- Packets that contain the Fragment header and the Packet Truncated option.
- Packets that contain the Truncation Eligible option and the Packet Truncated option.
- Packets that specify an Option Data Length greater than 0 in the Truncation Eligible option.

- Packets that specify an Option Data Length greater than 0 in the Truncated Packet option.

- Packets that have a total length less than the IPv6 minimum link MTU and contain the Packet Truncated option.

If an intermediate node cannot forward one of the above-mentioned packets because of an MTU issue, its behavior is as described in [RFC8200]. The intermediate node discards the packet and sends an ICMP PTB message to the source node. It does not truncate or forward the packet.

When the destination node receives one of the above-mentioned packets, it MUST:

- Discard the packet

- Send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the first invalid option.

The destination node MUST NOT send an ICMP PTB message.

11. Network Considerations

The procedures described herein rely upon the networks ability:

- To convey packets that contain destination options from the source node to the destination node.

- To convey ICMP Parameter Problem messages in the reverse direction.

Operational experience [RFC7872] reveals that a significant number of networks drop packets that contain IPv6 destination options. Likewise, many networks drop ICMP Parameter Problem messages.

[I-D.bonica-6man-unrecognized-opt] describes procedures that upper-layer protocols can execute to verify that the above-mentioned requirements are satisfied. Upper-layer protocols can execute these procedures before emitting packets that contain the Truncation Eligible option.

12. Encapsulating Security Payload Considerations

An IPv6 packet can contain both:

- An Encapsulating Security Payload (ESP) [RFC4303] header.

- Truncation options (i.e., the Truncation Eligible or Truncated Packet options).

In this case, the packet MUST contain a Destination Options header that precedes the ESP. That Destination Options header contains the truncation options and is not protected by the ESP. The packet MAY also contain another Destination Options header the follows the ESP. That Destination Options header is protected by the ESP and MUST NOT contain the truncation options.

As per RFC 4303, a packet can contain two Destination Options headers one preceding the ESP and one following the ESP.

13. Extension Header Considerations

According to [RFC8200], the following IPv6 extension headers can contain options:

- The Hop-by-hop Options header.

- The Destination Options header.

The Hop-by-hop option can be examined by each node along the path to a packet's destination. Destination options are examined by the destination node only. However, [RFC2473] provides a precedent for intermediate nodes examining the Destination options on an exception basis. (See the Tunnel Encapsulation Limit.)

The truncation options described herein are examined by:

- Intermediate nodes, on an exception basis (i.e, when the packet cannot be forwarded due to MTU issues).

- The Destination node.

Therefore, the above-mentioned options can be processed most efficiently when they are contained by the Destination Option header. When contained by the Destination Options header, the above-mentioned options are examined by intermediate nodes on an exception basis, only when they are relevant. If contained by the Hop-by-hop Options header, they are always examined by intermediate nodes, even when they are irrelevant.

14. Security Considerations

PMTUD is vulnerable to ICMP PTB forgery attacks. The procedures described herein do nothing to mitigate that vulnerability.

The procedures described herein are susceptible to a new variation on that attack, in which an attacker forges a truncated packet. In this case, the attackers cause the Destination Node to produce an ICMP PTB message on their behalf. To some degree, this vulnerability is mitigated, because the Destination Node will not emit an ICMP PTB message in response to a truncated packet whose length is less than the IPv6 minimum link MTU.

In order to mitigate denial of service attacks, intermediate nodes MUST rate limit the number of packets that they truncate per second.

15. IANA Considerations

IANA is requested to allocate the following codepoints from the Destination Options and Hop-by-hop Options registry (https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2).

- Truncation Eligible ("act-bits" are "00. "chg-bit" can be either 0 or 1.)
- Truncated Packet ("act-bits" are "10". "chg-but can be either 0 or 1.)

16. Acknowledgements

Special thanks to Mike Heard, Geoff Huston, Joel Jaeggli, Tom Jones, Andy Smith, Jinmei Tatuya, and Reji Thomas who reviewed and commented on this document.

17. References

17.1. Normative References

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

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

17.2. Informative References

[I-D.bonica-6man-unrecognized-opt] Bonica, R. and J. Leddy, "The IPv6 Probe Option", draft-bonica-6man-unrecognized-opt-03 (work in progress), August 2018.

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

[RFC7690] Byerly, M., Hite, M., and J. Jaeggli, "Close Encounters of the ICMP Type 2 Kind (Near Misses with ICMPv6 Packet Too Big (PTB))", RFC 7690, DOI 10.17487/RFC7690, January 2016, <https://www.rfc-editor.org/info/rfc7690>.

[RFC7872] Gont, F., Linkova, J., Chown, T., and W. Liu, "Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World", RFC 7872, DOI 10.17487/RFC7872, June 2016, <https://www.rfc-editor.org/info/rfc7872>.

Authors' Addresses

John Leddy
Unaffiliated

Email: john@leddy.net

Ron Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, Virginia 20171
USA

Email: rbonica@juniper.net

Igor Lubashev
Akamai Technologies
Cambridge, MA
USA

Email: ilubashe@akamai.com

IPv6 Maintenance
Internet-Draft
Intended status: Standards Track
Expires: April 4, 2019

L. Colitti
E. Kline
J. Linkova
Google

October 1, 2018

Discovering PREF64 in Router Advertisements
draft-pref64folks-6man-ra-pref64-02

Abstract

This document specifies a Router Advertisement option to communicate NAT64 prefixes to clients.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 4, 2019.

Copyright Notice

Table of Contents

1. Introduction

NAT64 [RFC6146] with DNS64 [RFC6147] is a widely-deployed mechanism to provide IPv4 access on IPv6-only networks. In order to support functions such as local validation of DNSSEC [RFC4033] responses, 464xlat [RFC6877], and local IPv4 address synthesis [RFC8305], the host must be aware of the NAT64 prefix in use by the network. This document specifies a Router Advertisement [RFC4861] option to communicate the NAT64 prefix to hosts.

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].

1.2. Terminology

Pref64: an IPv6 prefix used for IPv6 address synthesis [RFC6146];

RA Router Advertisement, a message used by IPv6 routers to advertise their presence together with various link and Internet parameters ([RFC4861]);

PvD-aware host A host that supports the association of network configuration information into PvDs and the use of these PvDs. Also named PvD-aware node in [RFC7556].

2. Why include the NAT64 prefix in Router Advertisements

Fate sharing: NAT64 requires a routing to be configured. IPv6 routing configuration requires receiving an IPv6 Router Advertisement [RFC4861]. Compared to currently-deployed NAT64 prefix discovery methods such as [RFC7050], including the NAT64 prefix in the Router Advertisement minimizes the number of packets required to configure a host. This speeds up the process of connecting to a network that supports NAT64/DNS64, and simplifies host implementation by removing the possibility that the host can have an incomplete layer 3 configuration (e.g., IPv6 addresses and prefixes, but no NAT64 prefix).

Deployability: all IPv6 hosts and networks are required to support [RFC4861]. Other options such as [RFC7225] require implementing other protocols.

3. Semantics

For simplicity, this option only supports a NAT64 prefix length of 96 bits, as this is by the most common configuration used by hosts. Networks using one of the other prefix lengths supported in ([RFC6052]) can use other mechanisms such as [RFC7050] or [RFC7225]. If different prefix lengths become common, another RA option can be created to configure them.

This option may appear more than once in a Router Advertisement. Host behaviour with regards to synthesizing IPv6 addresses from IPv4 addresses SHOULD follow the recommendations given in Section 3 of [RFC7050], limited to the NAT64 prefixes that have non-zero lifetime.

This option specifies exactly one NAT64 prefix for all IPv4 destinations. If the network operator desires to route different parts of the IPv4 address space to different NAT64 devices, this can be accomplished by routing more specifics of the NAT64 prefix to those devices. For example, if the operator would like to route 10.0.0.0/8 through NAT64 device A and the rest of the IPv4 space through NAT64 device B, and the operator’s NAT64 prefix is 2001:db8:a:b::/96, then the operator can route 2001:db8:a:b::a00:0/104 to NAT64 A and 2001:db8:a:b::/64 to NAT64 B.

In a network that provides both IPv4 and NAT64, it may be desirable for certain IPv4 addresses not to be translated. An example might be private address ranges that are local to the network and should not be reached through the NAT64. This type of configuration cannot be conveyed to hosts using this option, or through other NAT64 prefix provisioning mechanisms such as [RFC7050] or [RFC7225]. This problem does not apply in IPv6-only networks, because in such networks, the

host does not have an IPv4 address and cannot reach any IPv4 destinations without the NAT64.

4. Option format

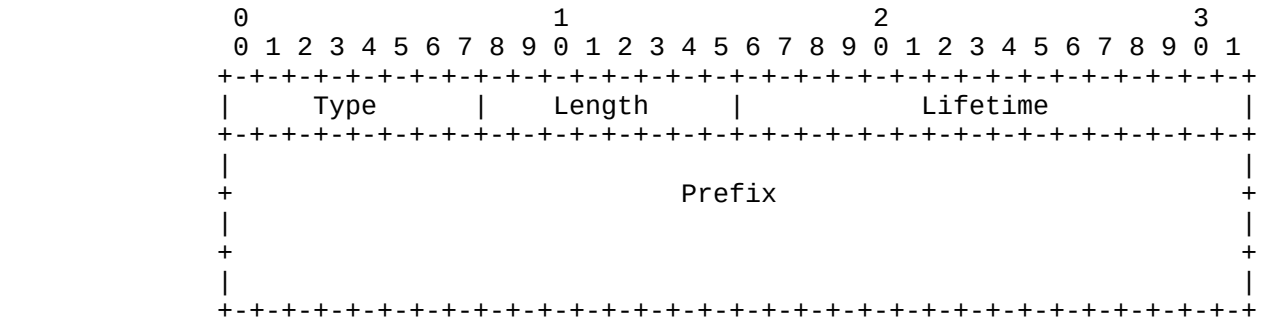


Figure 1: NAT64 Prefix Option Format

Fields:

Type 8-bit identifier of the RDNSS option type as assigned by IANA: TBD

Length 8-bit unsigned integer. The length of the option (including the Type and Length fields) is in units of 8 octets. The sender MUST set the Length to 2. A host MUST ignore the NAT64 prefix option if the length field value is 1. If the Length field value exceeds 2, the host MUST utilize the first 16 octets and ignore the rest of the option.

Lifetime 16-bit unsigned integer. The maximum time in seconds over which this NAT64 prefix MAY be used. The value of Lifetime SHOULD by default be set to lesser of 3 x MaxRtrAdvInterval or 65535 seconds. A value of zero means that the prefix MUST no longer be used.

Prefix The 96-bit NAT64 prefix.

5. Handling Multiple NAT64 Prefixes

In some cases a host may receive multiple NAT64 prefixes from different sources. Possible scenarios include (but are not limited to):

- the host is using multiple mechanisms to discover Pref64 prefixes (e.g. by using PCP ([RFC7225]) and/or by resolving IPv4-only fully

qualified domain name ([RFC7050]) in addition to receiving the Pref64 RA option);

- The pref64 option presents in a single RA more than once;
- the host receives multiple RAs with different Pref64 prefixes on one or multiple interfaces.

When multiple Pref64 were discovered via RA Pref64 Option (the Option presents more than once in a singe RA or multiple RAs were received), host behaviour with regards to synthesizing IPv6 addresses from IPv4 addresses SHOULD follow the recommendations given in Section 3 of [RFC7050], limited to the NAT64 prefixes that have non-zero lifetime..

When different Pref64 are discovered by using multiple mechanisms, hosts SHOULD select one source of infromation only. The RECOMMENDED order is:

- PCP-discovered prefixes ([RFC7225]), if supported;
- Pref64 discovered via RA Option;
- Pref64 resolving IPv4-only fully qualified domain name ([RFC7050])

Note that if the network provides Pref64 both via this RA option and [RFC7225], hosts that receive the Pref64 via RA option may choose to use it imediately before waiting for PCP to complete, and therefore some traffic may not reflect any more detailed configuration provided by PCP.

6. Multihoming

Like most IPv6 configuration information, the Pref64 option is specific to the network on which it is received. For example, a Pref64 option received on a particular wireless network may not be usable unless the traffic is also sourced on that network. Similarly, a host connected to a cellular network that povides NAT64 generally cannot use that NAT64 for destinations reached through a VPN tunnel that terminates outside that network.

Thus, correct use of this option on a multihomed host generally requires the host to be PVD-aware.

This issue is not specific to the Pref64 RA option and, for example, is quite typical for DNS resolving on multihomed hosts (e.g. a host might resolve a destination name by using the corporate DNS server

via the VPN tunnel but then send the traffic via its Internet-facing interface).

7. IANA Considerations

The IANA is requested to assign a new IPv6 Neighbor Discovery Option type for the PREF64 option defined in this document.

Option Name	Type
PREF64 option	(TBD)

Table 1

The IANA registry for these options is:

https://www.iana.org/assignments/icmpv6-parameters [1]

8. Security Considerations

Because Router Advertisements are required in all IPv6 configuration scenarios, on IPv6-only networks, Router Advertisements must already be secured, e.g., by deploying RA guard [RFC6105]. Providing all configuration in Router Advertisements increases security by ensuring that no other protocols can be abused by malicious attackers to provide hosts with invalid configuration.

The security measures that must already be in place to ensure that Router Advertisements are only received from legitimate sources eliminate the problem of NAT64 prefix validation described in section 3.1 of [RFC7050].

9. Acknowledgements

Thanks to the following people (in alphabetical order) for their review and feedback: Brian E Carpenter, Nick Heatley, David Schinazi.

10. References

10.1. Normative References

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

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <https://www.rfc-editor.org/info/rfc6052>.

10.2. Informative References

[I-D.ietf-intarea-provisioning-domains] Pfister, P., Vyncke, E., Pauly, T., Schinazi, D., and W. Shao, "Discovering Provisioning Domain Names and Data", draft-ietf-intarea-provisioning-domains-02 (work in progress), June 2018.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <https://www.rfc-editor.org/info/rfc6105>.

[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>.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <https://www.rfc-editor.org/info/rfc6147>.

[RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, DOI 10.17487/RFC6877, April 2013, <https://www.rfc-editor.org/info/rfc6877>.

[RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis", RFC 7050, DOI 10.17487/RFC7050, November 2013, <https://www.rfc-editor.org/info/rfc7050>.

[RFC7225] Boucadair, M., "Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol (PCP)", RFC 7225, DOI 10.17487/RFC7225, May 2014, <https://www.rfc-editor.org/info/rfc7225>.

[RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <https://www.rfc-editor.org/info/rfc7556>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/info/rfc8305>.

10.3. URIs

[1] https://www.iana.org/assignments/icmpv6-parameters

Authors' Addresses

Lorenzo Colitti
Google
Roppongi 6-10-1
Minato, Tokyo 106-6126
JP

Email: lorenzo@google.com

Erik Kline
Google
Roppongi 6-10-1
Minato, Tokyo 106-6126
JP

Email: ek@google.com

Jen Linkova
Google
1 Darling Island Rd
Pyrmont, NSW 2009
AU

Email: furry@google.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 1, 2019

O. Troan
Cisco Systems

September 28, 2018

Path MTU discovery solution space
draft-troan-6man-pmtu-solution-space-00

Abstract

Path MTU discovery has turned out to be a thorny problem that has haunted the Internet community for decades. Lately there has been some work both at the transport layer and at the network layer. This memo lists the solutions the author is aware of from the perspective of the network layer.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 1, 2019.

Copyright Notice

1. Introduction

Path MTU discovery has turned out to be harder than expeced. In IPv6 we set out following the same model as for IPv4. The sending host maintains a MTU cache, that is updated based on received ICMP PMTUD messages. That solution has a few short-comings:

- Sending of ICMP PMTUD messages is throttled in routers [RFC4443]
- It’s not efficient if links along the path have decreasingly smaller MTU, then multiple rounds of large packet, resulting ICMP PMTUD happens.
- ICMP might be ignored by host stacks / applications
- As ICMP looks different than application traffic, it might be blocked by routers.
- Doesn’t work well in an anycast scenario (but what does).

2. Requirements / Goals

1. Avoid MTU black-holes [RFC2923].
2. Detect the Path MTU in single round trip.
3. Adapt to varying MTU over the connection life time.
4. The signalling of the MTU back to the sender must be indistinguishable from application traffic to lessen risk of filtering.
5. Design a mechanism that ensures that neither MTU probes nor MTU signalling back to sender are more likely to be dropped than other application traffic.
6. Must be deployable and anchored in transport / application areas. Otherwise https://xkcd.com/927/
7. [Optional?] Support neighbors on the same link which support higher MTU than link MTU see [I-D.van-beijnum-multi-mtu]

3. Network layer solutions for Path MTU discovery

- PMTUD [RFC8201]
- On-path fragmentation, IPv4 style. We know this one.

- Packet truncation. [I-D.leddy-6man-truncate]. The source sets a truncation elligble flag in the packet, routers on the path may truncate f the packet is too big, and sets a truncated done flag. Then the receiver signals the learnt forward MTU back to the sender. Either via existing ICMP PMTUD or a transport layer option. This is an example of a solution which does not require the sender having to accept packets from intermediate nodes.

- MTU recording. Probe packets are sent, either as part of data packets, if those are guaranteed not to exceed MTU. Some trigger in the header (ECN like flags) or a HBH option is required for the router to record the smallest MTU along the path. Application / Transport would have to periodically include the probe trigger in data packets to detect changes in path MTU.

3.1. Common problems

How is the router along the path "triggered" to put this packet on the exception path? For current and the truncation scheme it’s a simple check in the forwarding path for the size of packet versus outgoing interface MTU. For e.g. a recording MTU mechanism it would have to be flags in the IPv6 header or an HBH option.

How should the forward path MTU be signalled back to the sender? The signal should look like any other application traffic to avoid filtering or is it sufficient to avoid sending from intermitent nodes.

4. Solutions at other layers

In addition there are solutions at the transport layer, that work in co-hort or independently of the network layer soltusions. [RFC4821] and [I-D.ietf-tsvwg-datagram-plpmtud].

One could also imagine other solutions, e.g. to include MTU in router advertisements in BGP, so that a BGP speaker could calculate the end to end MTU across the set of administrative domains.

5. Conclusion

What are our options? Even if we developed a new PMTU mechanism, IP stacks must deal with networks where the new mechanism isn’t yet deployed. Will a new mechanism be so much better that it provides enough value for it to be deployed? Or should we at the network layer just punt this to transport?

6. References

[I-D.ietf-tsvwg-datagram-plpmtud]
Fairhurst, G., Jones, T., Tuexen, M., and I. Ruengeler, "Packetization Layer Path MTU Discovery for Datagram Transports", draft-ietf-tsvwg-datagram-plpmtud-04 (work in progress), September 2018.

[I-D.leddy-6man-truncate]
Leddy, J. and R. Bonica, "IPv6 Packet Truncation", draft-leddy-6man-truncate-04 (work in progress), June 2018.

[I-D.van-beijnum-multi-mtu]
Beijnum, I., "Extensions for Multi-MTU Subnets", draft-van-beijnum-multi-mtu-05 (work in progress), March 2016.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, DOI 10.17487/RFC2923, September 2000, <https://www.rfc-editor.org/info/rfc2923>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/info/rfc4821>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

Author’s Address

Ole Troan
Cisco Systems
Philip Pedersens vei 1
Lysaker 1366
Norway

Email: ot@cisco.com