

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.

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."

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the

document authors. All rights reserved.

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

Table of Contents

1. Introduction.....	3
2. Conventions Used in This Document.....	3
2.1. Abbreviations.....	3
2.2. Terminology and Reference Topology.....	4
3. OAM Building Blocks.....	5
3.1. O-flag in Segment Routing Header.....	5
3.2. OAM Segments.....	7

3.2.1. End.OP: OAM Endpoint with Punt.....	7
3.2.2. End.OTP: OAM Endpoint with Timestamp and Punt.....	8
4. OAM Mechanisms.....	8
4.1. Ping.....	9
4.1.1. Classic Ping.....	9
4.1.2. Pinging a SID Function.....	10
4.1.2.1. End-to-end ping using END.OP/ END.OTP.....	11
4.1.2.2. Segment-by-segment ping using O-flag (Proof of Transit).....	11
4.2. Error Reporting.....	13
4.3. Traceroute.....	13
4.3.1. Classic Traceroute.....	13
4.3.2. Traceroute to a SID Function.....	15
4.3.2.1. Hop-by-hop traceroute using END.OP/ END.OTP....	16
4.3.2.2. Tracing SRv6 Overlay.....	17
4.4. Monitoring of SRv6 Paths.....	19
5. Security Considerations.....	20
6. IANA Considerations.....	20
6.1. ICMPv6 type Numbers Registry.....	20
7. References.....	21
7.1. Normative References.....	21
7.2. Informative References.....	22
8. Acknowledgments.....	22

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.

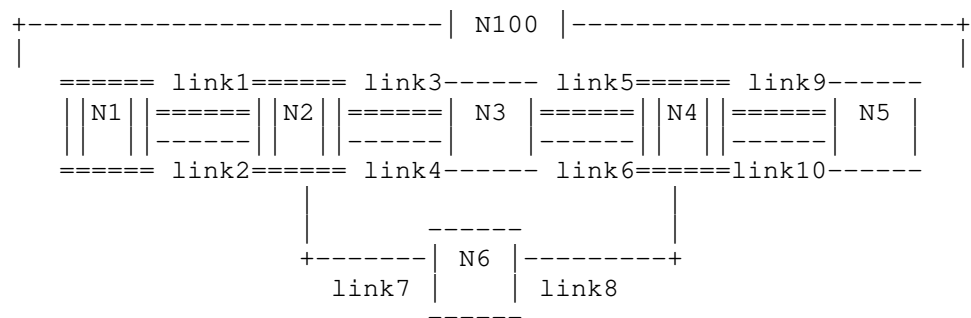


Figure 1 Reference Topology

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 responds 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 responds 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 responds 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.0 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.0=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 responds 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 responds 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, et 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

6man
Internet-Draft
Intended status: Standards Track
Expires: February 11, 2019

R. Bonica
Juniper Networks
J. Leddy
Comcast
August 10, 2018

The IPv6 Probe Option
draft-bonica-6man-unrecognized-opt-03

Abstract

This document defines a new IPv6 option, called the Probe option. The Probe option elicits an ICMPv6 Parameter Problem message from all nodes that process it. When a node sends a packet that contains the Probe option and receives an ICMPv6 Parameter Problem message in response, it has verified the network's ability to convey packets that contain the Probe option.

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 February 11, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. The Probe Option	4
4. Discovering Network Capabilities	5
5. Security Considerations	6
6. IANA Considerations	6
7. Acknowledgements	6
8. References	6
8.1. Normative References	6
8.2. Informative References	7
Authors' Addresses	7

1. Introduction

In IPv6 [RFC8200], optional internet-layer information is encoded in extension headers. Two extension headers, the Hop-by-Hop Options header and the Destination Options header, contain a variable number of options. Each option contains the following fields:

- o Option Type
- o Opt Data Length
- o Option Data

The Option Type identifiers are encoded so that their highest-order 2 bits specify the action to be taken if the processing node does not recognize the option. Encodings follow:

- o 00 - Skip over the option and continue processing the header.
- o 01 - Discard the packet.
- o 10 - Discard the packet and send an ICMPv6 [RFC4443] Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.
- o 11 - Discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMPv6 Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

Several upper-layer protocols [RFC6275] [I-D.ledly-6man-truncate] emit packets that contain IPv6 destination options. These protocols rely the network to convey packets that contain the IPv6 Destination Options header.

A subset of those protocols emit IPv6 destination options with high-order bits equal to "10" and "11". These IPv6 destination options elicit ICMPv6 Parameter Problem messages from destination nodes that do not recognize them. The above-mentioned protocols perform better when the network can convey ICMPv6 Parameter Problem messages from the destination node to the source node.

Operational experience [RFC7872] reveals that a significant number of networks drop all packets that contain the IPv6 Destination Options header. Similarly, a significant number of networks allow packets that contain the IPv6 Destination Options header, but only if Destination Options header does not exceed a specific size. Finally, many networks drop all ICMP Parameter Problem messages.

This document describes procedures by which a source node can discover relevant capabilities of the network that connects it to a destination node. Using these procedures, the source node can determine:

- o Whether the network can convey a packet containing a Destination Options header of a specific size from the source node to a destination node.
- o Whether the network can convey an ICMPv6 Parameter Problem message from the destination node to the source node.

In order to support the above-mentioned procedures, this document defines a new IPv6 option, called the Probe option. The Probe option elicits an ICMPv6 Parameter Problem message from all nodes that process it. It elicits an IPv6 Parameter Problem message, regardless of whether the processing node recognizes the option. When a source node sends a packet that contains the Probe option and receives an ICMPv6 Parameter Problem message in response, it has verified the above-mentioned network capabilities.

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. The Probe Option

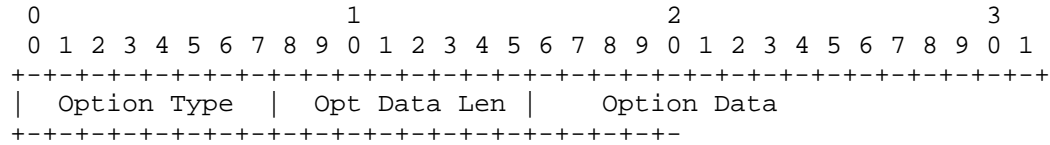


Figure 1

Figure 1 depicts the Probe Option.

Option fields are as follows:

- o Option Type - Probe Option. Value TBD by IANA. See Notes below.
- o Opt Data Len - Length of Option Data, measured in bytes.
- o Option Data - MUST be set to zero on transmission. MUST be ignored on receipt.

The Opt Data Len and Option Data fields can be used to expand the Probe Option and the Destination Options header that contains it to a required length. See Section 4 for details.

A packet MAY contain multiple instances of the Probe option. In IPv6, the maximum size of a Destination Options header is 2048 bytes, while the maximum size of an option instance is only 256 bytes. Therefore, multiple instances of the Probe option are required to expand the Destination Options header beyond 256 bytes.

All nodes process the Probe option as follows, regardless of whether they recognize the option:

- o Discard the packet.
- o Send an ICMPv6 Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

NOTE 1: The highest-order two bits of the Option Type (i.e., the "act" bits) are 10. These bits specify the action taken by a destination node that does not recognize Probe option. The required action is to discard the packet and send an ICMPv6 Parameter Problem, Code 2, message to the packet's Source Address, pointing to the Probe Option Type.

NOTE 2: The third highest-order bit of the Option Type (i.e., the "chg" bit) is 0. This indicates that Option Data cannot be modified along the path between the packet's source and its destination.

4. Discovering Network Capabilities

Assume that a source node needs to determine whether the network can convey a packet from itself to a destination node. The packet contains a Destination Options header whose length is N bytes. As per [RFC8200], the Destination Options header length must be a multiple of 8. Therefore, N must be a multiple of 8.

The source node executes the following procedure:

- o Set a short timer (e.g., one or two seconds).
- o Send a probe packet.
- o Wait for either a) an ICMPv6 Parameter Problem message that matches the probe packet, or b) timer expiration

The probe packet contains an IPv6 Destination Options header and the IPv6 Destination Options header contains one or more instances of Probe option. The number of Probe option instances and the length of Option Data in each instance are chosen so that the Destination Options header length will be equal to N.

In order to influence how the packet is routed to its destination, the probe packet MAY contain upper-layer headers. However, because the packet contains the Probe option, it is always discarded and is never delivered to an upper-layer protocol.

An ICMPv6 Parameter Problem message matches a probe packet if the initial bytes of the probe packet appear in the ICMP Parameter Problem message.

If the source node receives an ICMP Parameter Problem message that matches the probe, both of the following statements are true:

- o The network can convey a packet containing a Destination Options header of a specific size from the source node to a destination node.
- o The network can convey an ICMPv6 Parameter Problem message from the destination node to the source node.

If the timer expires, at least one of the following statements is true:

- o The network cannot convey a packet containing a Destination Options header of a specific size from the source node to a destination node.
- o The network cannot convey an ICMPv6 Parameter Problem message from the destination node to the source node.
- o Either the probe or the ICMPv6 Parameter Problem message was lost due to a transient issue (e.g., congestion).

As noted above, transient issues can cause false negative results. Therefore, this procedure MAY be repeated after initial failure.

5. Security Considerations

This document introduces no new security vulnerabilities. Any security vulnerabilities exposed by the Probe option are currently exposed by all undefined or unrecognized option types. This is because the Probe option elicits the same behavior as an undefined or unrecognized option

6. IANA Considerations

IANA is requested to allocate a codepoint from the Destination Options and Hop-by-hop Options registry (<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2>). This option is called "Probe". The "act" bits are 10 and the "chg" bit is 0.

7. Acknowledgements

Thanks to Ross Callon, Fernando Gont and Jinmei Tatuya for their careful review of this document.

8. References

8.1. 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>>.
- [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>>.

8.2. Informative References

- [I-D.leddy-6man-truncate] Leddy, J. and R. Bonica, "IPv6 Packet Truncation", draft-leddy-6man-truncate-04 (work in progress), June 2018.
- [RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <<https://www.rfc-editor.org/info/rfc6275>>.
- [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

Ron Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, Virginia 20171
USA

Email: rbonica@juniper.net

John Leddy
Comcast
1717 John F Kennedy Blvd.
Philadelphia, PA 19103
USA

Email: john_leddy@comcast.com

6man
Internet-Draft
Intended status: Standards Track
Expires: 21 July 2022

R. Bonica
Juniper Networks
Y. Kamite
NTT Communications Corporation
L. Jalil
Verizon
Y. Zhou
ByteDance
G. Chen
Baidu
17 January 2022

The IPv6 Tunnel Payload Forwarding (TPF) Option
draft-bonica-6man-vpn-dest-opt-17

Abstract

This document explains how IPv6 options can be used in IPv6 tunnels.
It also defines the IPv6 Tunnel Payload Forwarding (TPF) option.

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 21 July 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the
document authors. All rights reserved.

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

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	4
3. The IPv6 Tunnel Payload Forwarding (TPF) Option	4
4. TPF Information Determines Next-Protocol Engine Behavior . .	5
5. TPF Information Semantics	5
6. Virtual Private Networking (VPN) Applications	6
7. Security Considerations	6
8. IANA Considerations	6
9. Acknowledgements	7
10. Contributors	7
11. References	7
11.1. Normative References	7
11.2. Informative References	8
Authors' Addresses	8

1. Introduction

This document explains how IPv6 options [RFC8200] can be used in IPv6 tunnels. It also defines the IPv6 Tunnel Payload Forwarding (TPF) option.

An IPv6 tunnel [RFC2473] connects two nodes, called the entry-point and the exit-point. The entry-point receives a packet and encapsulates it in a Tunnel IPv6 Header. Figure 1 depicts the encapsulation.

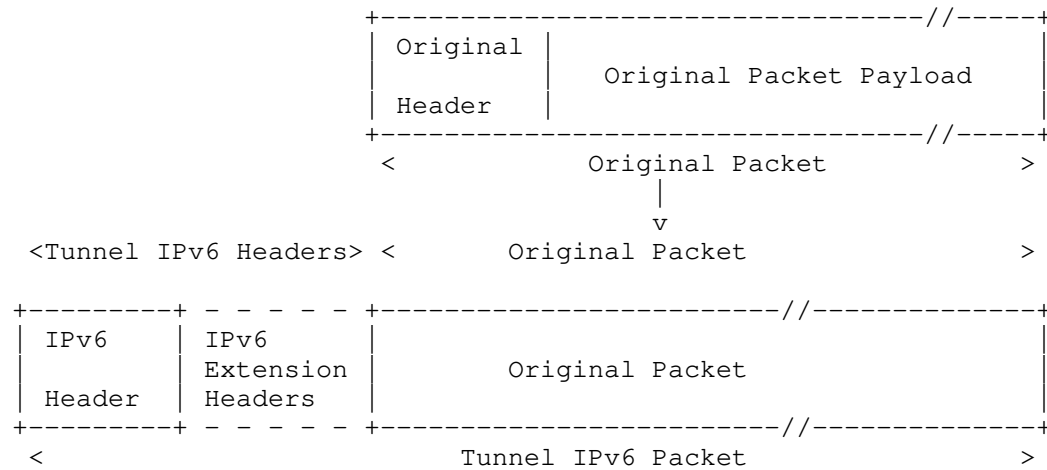


Figure 1: IPv6 Tunnel Encapsulation

The original packet can be any layer-2 or layer-3 packet (e.g., Ethernet, IPv4, IPv6). The Tunnel Header is an IPv6 header followed by zero or more extension headers. The resulting packet is a Tunnel IPv6 Packet.

The entry-point sends the Tunnel IPv6 Packet to the exit-point which then executes the following procedure:

- * Process the Tunnel IPv6 Header.
- * Remove the Tunnel IPv6 Header, exposing the original packet.
- * Submit the original packet to the next-protocol engine.

The exit-point node processes the Tunnel IPv6 Header in strict left-to-right order. It processes the IPv6 header first and then processes extension headers in the order that they appear in the packet. The IPv6 header, and each extension header, includes a Next Header field. The last Next Header field processed identifies the next-protocol engine.

Entry-point nodes can send optional information to the next-protocol engine on the exit-point node. For example, the entry-point can indicate:

- * The interface through which the next-protocol engine should send the packet.

- * The routing table that the next-protocol engine should use to process the packet.

To send this information, the entry-point node includes an IPv6 Destination Option header in the Tunnel IPv6 Header. The IPv6 Destination Options header includes an IPv6 TPF option and the IPv6 TPF option includes TPF information. The next-protocol engine on the exit-point node uses TPF information when it forwards the original packet.

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. The IPv6 Tunnel Payload Forwarding (TPF) Option

The TPF Option contains the following fields:

- * Option Type: 8-bit selector. TPF option. Value TBD by IANA. (Suggested value: 0x41). See Note below.
- * Opt Data Len - 8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Option Length fields. This field MUST be set to 4.
- * Option Data - 32-bits. Tunnel Payload Forwarding (TPF) Information.

The TPF option MAY appear in a Destination Options header that precedes an upper-layer header. It MUST NOT appear in a Hop-by-hop Options header or in a Destination Options header that precedes a Routing header.

NOTE : The highest-order two bits of the Option Type (i.e., the "act" bits) are 01. These bits specify the action taken by a destination node that does not recognize the option. The required action is to discard the packet. The third highest-order bit of the Option Type (i.e., the "chg" bit) is 0. This indicates that Option Data cannot be modified along the path between the packet's source and its destination.

4. TPF Information Determines Next-Protocol Engine Behavior

An exit-point node supports one or more next-protocol engines (e.g., Ethernet, IPv4, IPv6). Each next-protocol engine supports a default forwarding procedure and zero or more special forwarding procedures.

When an exit-point node submits a packet to a next-protocol engine without TPF information, the next-protocol engine executes its default forwarding procedure. For example, assume that the exit-point node receives the following Tunnel IPv6 Packet:

- * The Tunnel IPv6 Packet does not contain TPF information.
- * The original packet is IPv4.

In this case, the exit-point node processes and removes the Tunnel IPv6 Header. It then submits the original packet, without any TPF information, to the IPv4 protocol engine.

The IPv4 protocol engine executes its default forwarding procedure. It searches its Forwarding Information Base (FIB) for an entry that matches the original packet's destination address. If the search returns a FIB entry, the protocol engine forwards the packet through an interface that the FIB entry identifies.

When an exit-point node submits a packet to a next-protocol engine with TPF information, the next-protocol engine executes a special forwarding procedure. For example, assume that the exit-point node receives the following Tunnel IPv6 packet:

- * The Tunnel IPv6 Packet contains TPF information that identifies an interface.
- * The original packet is IPv4.

In this case, the exit-point node processes and removes the Tunnel IPv6 Header. It then submits the original packet, along with TPF information, to the IPv4 protocol engine.

The IPv4 protocol engine executes a special forwarding procedure. It forwards the packet through the interface identified by TPF information, without searching the FIB.

5. TPF Information Semantics

TPF information is opaque. While it must be understood by the entry-point node and the exit-point node, it does not need to be understood by any other node.

6. Virtual Private Networking (VPN) Applications

The IPv6 TPF option is useful in deployments where IPv6 tunnels carry:

- * Layer 3 Virtual Private Network (L3VPN) [RFC4364] traffic.
- * Ethernet Virtual Private Network (EVPN) [RFC7432] traffic.

When an IPv6 tunnel carries L3VPN traffic, VPN context information can be encoded in an IPv6 TPF option. Therefore, the MPLS service label that is normally present in an L3VPN packet can be eliminated.

When an IPv6 tunnel carries EVPN traffic, VPN context information can be encoded in an IPv6 TPF option. Therefore, the UDP and VXLAN headers that might otherwise be present can be eliminated.

7. Security Considerations

TPF information MUST NOT be accepted from untrusted sources. The following are acceptable methods of risk mitigation:

- * Authenticate the IPv6 TPF option using the IPv6 Authentication Header (AH) [RFC4302] or the IPv6 Encapsulating Security Payload (ESP) Header [RFC4303].
- * Maintain a secure TPF domain.

All nodes at the edge of a secure TPF domain discard packets that satisfy the following criteria:

- * Contain an IPv6 TPF option.
- * Contain an IPv6 Destination Address that represents an interface inside of the secure TPF domain.

8. IANA Considerations

IANA is requested to allocate a code point from the Destination Options and Hop-by-hop Options registry (<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2>). This option is called "Tunnel Payload Forwarding Option". The "act" bits are 01 and the "chg" bit is 0. The suggested value is 0x41.

9. Acknowledgements

Thanks to Dr. Vanessa Ameen, Brian Carpenter, Adrian Farrel, Ishaan Gandhi, Tom Herbert, John Leddy, Srihari Sangli and Tony Li for their comments.

10. Contributors

Chris Lenart

Verizon

22001 Loudoun County Parkway

Ashburn, Virginia 20147 USA

Email: chris.lenart@verizon.com

Greg Presbury

Hughes Network Systems

11717 Exploration Lane

Germantown, Maryland 20876 USA

Email: greg.presbury@hughes.com

11. References

11.1. 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>>.
- [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>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.

- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [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>>.

11.2. Informative References

- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.

Authors' Addresses

Ron Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, Virginia 20171
United States of America

Email: rbonica@juniper.net

Yuji Kamite
NTT Communications Corporation
3-4-1 Shibaura, Minato-ku,
108-8118
Japan

Email: y.kamite@ntt.com

Luay Jalil
Verizon
Richardson, Texas
United States of America

Email: luay.jalil@one.verizon.com

Yifeng Zhou
ByteDance
Building 1, AVIC Plaza, 43 N 3rd Ring W Rd Haidian District
Beijing
100000
P.R. China

Email: yifeng.zhou@bytedance.com

Gang Chen
Baidu
No.10 Xibeiwang East Road Haidian District
Beijing
100193
P.R. China

Email: phdgang@gmail.com

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 16, 2019

W. Feng
PSU
D. Moses
Intel
September 12, 2018

Router Advertisement Prefix Option Extension for On-Demand Mobility
draft-feng-dmm-ra-prefixtype-03

Abstract

Router Advertisement / Router Solicitation is one of the ways for hosts to establish network IPv6 connectivity configuration. This document describes two approaches to allowing a router to specify mobility service type availability to mobile hosts. Mobile hosts can then configure their IP address to the preferred type of mobile connectivity. Two possibilities are considered: (i) creating an extension to the router advertisement prefix information option to allow the router to specify mobility connectivity types, and (ii) specifying a new RA options that allows the router to specify the mobility connectivity types.

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 March 16, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Notational Conventions	2
3. Router Advertisement Extensions	3
3.1. Modifying PIO	3
3.2. Adding a new RA option	5
4. Security Considerations	7
5. IANA Considerations	7
6. References	7
6.1. Normative References	7
6.2. Informative References	8
Authors' Addresses	8

1. Introduction

[I-D.ietf-dmm-ondemand-mobility] defines different types of mobility related network services provided by access network to mobile hosts. In particular, it defines different types of prefix continuity types as mobile nodes move between different points of attachments.

This document proposes two such options to the router advertisement message ([RFC4861]) to allow the router to convey mobility services associated with an Ipv6 prefix. The possibilities considered are: (i) creating an extension to the router advertisement prefix information option to allow the router to specify mobility connectivity types, and (ii) specifying a new RA options that allows the router to specify the mobility connectivity types.

For (i), the prefix information option is extended to support the specification of mobility type. In (ii), a new RA option field is provided to do the same.

2. Notational Conventions

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

3. Router Advertisement Extensions

IP prefixes are conveyed in Router Advertisement messages through the Prefix Information Option field ([RFC4861]). These prefix information option fields are used to allow hosts to configure their IPv6 addresses.

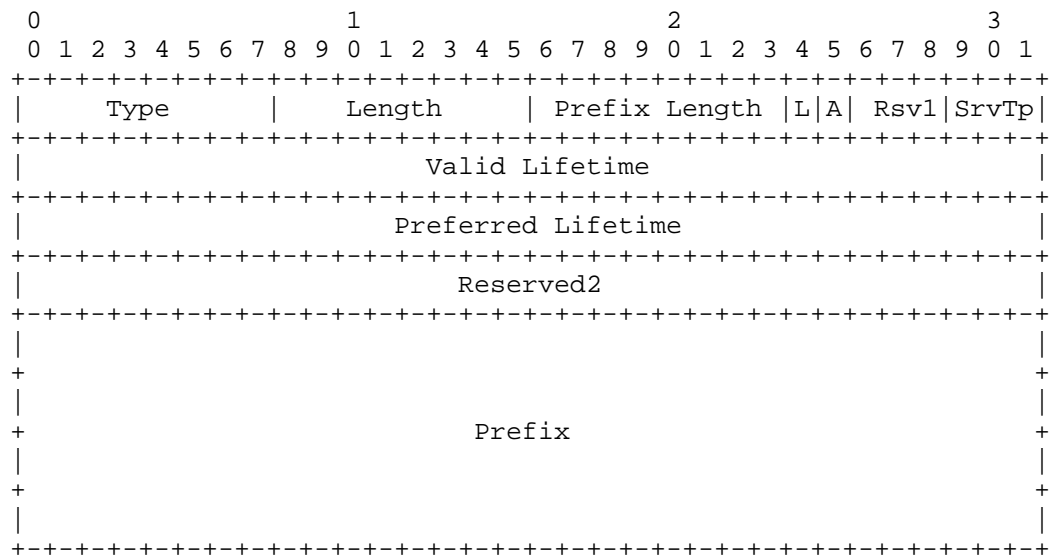
For distributed mobility management, there is a need for a network to be able to convey different prefixes for different connectivity scenarios. [I-D.ietf-dmm-ondemand-mobility] defines different service continuity requirements including: Non-Persistent, Session-Lasting, Fixed, and Graceful-replacement. Currently, however, there is no way for a router to specify the continuity type through a router advertisement message.

This document proposes two possibilities for modifying the router advertisement message to include mobility service options that it is offering to mobile hosts that are attached: (i) creating an extension to the router advertisement prefix information option (PIO) to allow the router to specify mobility connectivity types, and (ii) specifying a new RA options that allows the router to specify the mobility connectivity types.

3.1. Modifying PIO

The first option is to modify the PIO. The advantages of this approach are that it is semantically in line with the intended function. That is, specifying prefix options. This, however, requires the modification of several bits in the existing PIO to support the specification of the type.

The modified prefix information option fields are shown in the following figure:



Fields:

Type	3
Length	4
Prefix Length	8-bit unsigned integer. The number of leading bits in the Prefix that are valid. The value ranges from 0 to 128.
L	1-bit on-link flag. When set, indicates that this prefix can be used for on-link determination.
A	1-bit autonomous address-configuration flag. When set indicates that this prefix can be used for stateless address configuration.
Rsv1	3-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
SrvTp	3-bit field that specifies the service type. The field can have the following values:

Non-Persistent - a non-persistent IP prefix (1)

Session-Lasting - a session-lasting IP prefix (2)

Fixed - a fixed IP prefix (3)

Graceful-replacement - a graceful-replacement IP prefix (4)

The definition of these service types is available in [I-D.ietf-dmm-ondemand-mobility].

0 is reserved and should not be used. All other values (5-7) are reserved for future use.

The value of the Service Type indicates the type of continuity service committed by the network for the associated IPv6 prefix.

Once an IPv6 prefix type is provided, any subsequent messages involving this prefix (lease renewal - for example) must include the IPv6 Continuity Service option with the same service type that was assigned by the server during the initial allocation.

Given the list of IPv6 prefixes and their associated mobility service type, the mobile host can then configure its IP address to the appropriate service required by the application

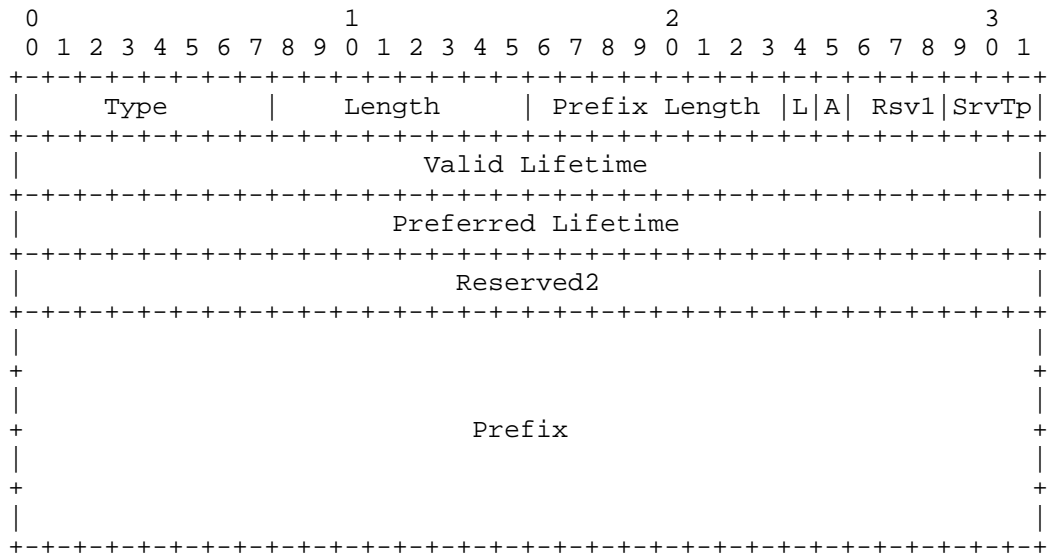
Mobile hosts that do not support this new option should ignore the prefix information option.

Routers should also send an additional prefix information option without the session-type field from time to time for hosts that do not support this new format.

3.2. Adding a new RA option

The second approach is to add a new RA option alongside the existing PIO (and other RA options). The advantage of this approach are that it leaves the existing PIO untouched. Furthermore, hosts that receive this option with the type that they do not understand can simply disregard it.

The new RA option specification is shown in the following figure:



Fields:

Type	Need to define new Type #
Length	4
Prefix Length	8-bit unsigned integer. The number of leading bits in the Prefix that are valid. The value ranges from 0 to 128.
L	1-bit on-link flag. When set, indicates that this prefix can be used for on-link determination.
A	1-bit autonomous address-configuration flag. When set indicates that this prefix can be used for stateless address configuration.
Rsv1	3-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
SrvTp	3-bit field that specifies the service type. The field can have the following values:

Non-Persistent - a non-persistent IP prefix (1)

Session-Lasting - a session-lasting IP prefix (2)

Fixed - a fixed IP prefix (3)

Graceful-replacement - a graceful-replacement IP prefix (4)

The definition of these service types is available in [I-D.ietf-dmm-ondemand-mobility].

0 is reserved and should not be used. All other values (5-7) are reserved for future use.

The value of the Service Type indicates the type of continuity service committed by the network for the associated IPv6 prefix.

Once an IPv6 prefix type is provided, any subsequent messages involving this prefix (lease renewal - for example) must include the IPv6 Continuity Service option with the same service type that was assigned by the server during the initial allocation.

Given the list of IPv6 prefixes and their associated mobility service type, the mobile host can then configure its IP address to the appropriate service required by the application

Mobile hosts that do not support this new option should ignore the prefix information option.

Routers should also send an additional prefix information option without the session-type field from time to time for hosts that do not support this new format.

4. Security Considerations

There are no specific security considerations for this option.

5. IANA Considerations

TBD

6. References

6.1. 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>>.

6.2. Informative References

- [I-D.ietf-dmm-distributed-mobility-anchoring]
Chan, A., Wei, X., Lee, J., Jeon, S., and C. Bernardos,
"Distributed Mobility Anchoring", draft-ietf-dmm-
distributed-mobility-anchoring-11 (work in progress),
August 2018.
- [I-D.ietf-dmm-ondemand-mobility]
Yegin, A., Moses, D., Kweon, K., Lee, J., Park, J., and S.
Jeon, "On Demand Mobility Management", draft-ietf-dmm-
ondemand-mobility-15 (work in progress), July 2018.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins,
C., and M. Carney, "Dynamic Host Configuration Protocol
for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July
2003, <<https://www.rfc-editor.org/info/rfc3315>>.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic
Host Configuration Protocol (DHCP) version 6", RFC 3633,
DOI 10.17487/RFC3633, December 2003,
<<https://www.rfc-editor.org/info/rfc3633>>.
- [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>>.
- [RFC7934] Colitti, L., Cerf, V., Cheshire, S., and D. Schinazi,
"Host Address Availability Recommendations", BCP 204,
RFC 7934, DOI 10.17487/RFC7934, July 2016,
<<https://www.rfc-editor.org/info/rfc7934>>.

Authors' Addresses

Wu-chi Feng
Portland State Univ.
Hillsboro
USA

Email: wuchi@pdx.edu

Internet-Draft Router Advertisement Prefix Option Extension September 2018

Danny Moses
Intel
Petah Tikva
Israel

Email: danny.moses@intel.com

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.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	4
3. Applicability Statements	4
4. IPv6-Only Definition	5
5. IPv6-Only Flag	5
6. Router and Operational Considerations	6
7. Host Behavior Considerations	7
8. IANA Considerations	8
9. Security Considerations	8
10. Acknowledgments	9
11. Change log [RFC Editor: Please remove]	9
12. References	12
12.1. Normative References	12
12.2. Informative References	13
Appendix A. Implementaton Status [RFC Editor: Please remove] . .	14
A.1. FreeBSD Implementation	14
A.2. Test using Scapy	14
Authors' Addresses	15

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:

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

- o Switches in multi-segment wireless networks may create IPv4 state for dual stack hosts (in particular, ARP cache entries to support ARP proxying).
- o 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.
- o Similarly, hosts may waste battery power on futile attempts to access services by sending IPv4 packets.
- o 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:

```

  0 1 2 3 4 5 6 7
+---+---+---+---+---+---+
|M|O|H|Prf|P|R|R|
+---+---+---+---+---+---+

```

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:

- o 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.
- o 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.
- o 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 title 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 intended 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

IPv6 Maintenance (6man) Working Group
Internet-Draft
Obsoletes: 4941 (if approved)
Intended status: Standards Track
Expires: May 6, 2021

F. Gont
SI6 Networks
S. Krishnan
Kaloom
T. Narten

R. Draves
Microsoft Research
November 2, 2020

Temporary Address Extensions for Stateless Address Autoconfiguration in
IPv6
draft-ietf-6man-rfc4941bis-12

Abstract

This document describes an extension to IPv6 Stateless Address Autoconfiguration that causes hosts to generate global scope addresses with randomized interface identifiers that change over time. Changing global scope addresses over time limits the window of time during which eavesdroppers and other information collectors may trivially perform address-based network activity correlation when the same address is employed for multiple transactions by the same host. Additionally, it reduces the window of exposure of a host as being accessible via an address that becomes revealed as a result of active communication. This document obsoletes RFC4941.

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 May 6, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Terminology	3
1.2. Problem Statement	4
2. Background	4
2.1. Extended Use of the Same Identifier	4
2.2. Possible Approaches	6
3. Protocol Description	6
3.1. Design Guidelines	7
3.2. Assumptions	7
3.3. Generation of Randomized Interface Identifiers	8
3.3.1. Simple Randomized Interface Identifiers	8
3.3.2. Hash-based Generation of Randomized Interface Identifiers	9
3.4. Generating Temporary Addresses	11
3.5. Expiration of Temporary Addresses	12
3.6. Regeneration of Temporary Addresses	13
3.7. Implementation Considerations	14
3.8. Defined Constants and Configuration Variables	14
4. Implications of Changing Interface Identifiers	15
5. Significant Changes from RFC4941	17
6. Future Work	18
7. Implementation Status	19
8. IANA Considerations	19
9. Security Considerations	19
10. Acknowledgments	20
11. References	21
11.1. Normative References	21
11.2. Informative References	22
Authors' Addresses	24

1. Introduction

[RFC4862] specifies "Stateless Address Autoconfiguration (SLAAC) for IPv6", which typically results in hosts configuring one or more "stable" IPv6 addresses composed of a network prefix advertised by a local router and a locally-generated Interface Identifier (IID). The security and privacy implications of such addresses have been discussed in detail in [RFC7721], [RFC7217], and [RFC7707]. This document specifies an extension for SLAAC to generate temporary addresses, that can help mitigate some of the aforementioned issues. This is a revision of RFC4941, and formally obsoletes RFC4941. Section 5 describes the changes from [RFC4941].

The default address selection for IPv6 has been specified in [RFC6724]. The determination as to whether to use stable versus temporary addresses can in some cases only be made by an application. For example, some applications may always want to use temporary addresses, while others may want to use them only in some circumstances or not at all. An Application Programming Interface (API) such as that specified in [RFC5014] can enable individual applications to indicate a preference for the use of temporary addresses.

Section 2 provides background information. Section 3 describes a procedure for generating temporary addresses. Section 4 discusses implications of changing interface identifiers (IIDs). Section 5 describes the changes from [RFC4941].

1.1. Terminology

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

The terms "public address", "stable address", "temporary address", "constant IID", "stable IID", and "temporary IID" are to be interpreted as specified in [RFC7721].

The term "global scope addresses" is used in this document to collectively refer to "Global unicast addresses" as defined in [RFC4291] and "Unique local addresses" as defined in [RFC4193], and not to "globally reachable" addresses, as defined in [RFC8190].

1.2. Problem Statement

Addresses generated using stateless address autoconfiguration [RFC4862] contain an embedded interface identifier, which may remain stable over time. Anytime a fixed identifier is used in multiple contexts, it becomes possible to correlate seemingly unrelated activity using this identifier.

The correlation can be performed by

- o An attacker who is in the path between the host in question and the peer(s) to which it is communicating, and who can view the IPv6 addresses present in the datagrams.
- o An attacker who can access the communication logs of the peers with which the host has communicated.

Since the identifier is embedded within the IPv6 address, it cannot be hidden. This document proposes a solution to this issue by generating interface identifiers that vary over time.

Note that an attacker, who is on path, may be able to perform significant correlation based on:

- o The payload contents of unencrypted packets on the wire
- o The characteristics of the packets such as packet size and timing

Use of temporary addresses will not prevent such correlation, nor will it prevent an on-link observer (e.g. the host's default router) from tracking all the host's addresses.

2. Background

This section discusses the problem in more detail, provides context for evaluating the significance of the concerns in specific environments, and makes comparisons with existing practices.

2.1. Extended Use of the Same Identifier

The use of a non-changing interface identifier to form addresses is a specific instance of the more general case where a constant identifier is reused over an extended period of time and in multiple independent activities. Any time the same identifier is used in multiple contexts, it becomes possible for that identifier to be used to correlate seemingly unrelated activity. For example, a network sniffer placed strategically on a link across which all traffic to/from a particular host crosses could keep track of which destinations

a host communicated with and at what times. Such information can in some cases be used to infer things, such as what hours an employee was active, when someone is at home, etc. Although it might appear that changing an address regularly in such environments would be desirable to lessen privacy concerns, it should be noted that the network prefix portion of an address also serves as a constant identifier. All hosts at, say, a home, would have the same network prefix, which identifies the topological location of those hosts. This has implications for privacy, though not at the same granularity as the concern that this document addresses. Specifically, all hosts within a home could be grouped together for the purposes of collecting information. If the network contains a very small number of hosts, say, just one, changing just the interface identifier will not enhance privacy, since the prefix serves as a constant identifier.

One of the requirements for correlating seemingly unrelated activities is the use (and reuse) of an identifier that is recognizable over time within different contexts. IP addresses provide one obvious example, but there are more. For example,

- o Many hosts also have DNS names associated with their addresses, in which case the DNS name serves as a similar identifier. Although the DNS name associated with an address is more work to obtain (it may require a DNS query), the information is often readily available. In such cases, changing the address on a host over time would do little to address the concerns raised in this document, unless the DNS name is changed at the same time as well (see Section 4).
- o Web browsers and servers typically exchange "cookies" with each other [RFC6265]. Cookies allow web servers to correlate a current activity with a previous activity. One common usage is to send back targeted advertising to a user by using the cookie supplied by the browser to identify what earlier queries had been made (e.g., for what type of information). Based on the earlier queries, advertisements can be targeted to match the (assumed) interests of the end-user.

The use of a constant identifier within an address is of special concern because addresses are a fundamental requirement of communication and cannot easily be hidden from eavesdroppers and other parties. Even when higher layers encrypt their payloads, addresses in packet headers appear in the clear. Consequently, if a mobile host (e.g., laptop) accessed the network from several different locations, an eavesdropper might be able to track the movement of that mobile host from place to place, even if the upper layer payloads were encrypted.

Changing global scope addresses over time limits the time window over which eavesdroppers and other information collectors may trivially correlate network activity when the same address is employed for multiple transactions by the same host. Additionally, it reduces the window of exposure of a host as being accessible via an address that becomes revealed as a result of active communication.

The security and privacy implications of IPv6 addresses are discussed in detail in [RFC7721], [RFC7707], and [RFC7217].

2.2. Possible Approaches

One approach, compatible with the stateless address autoconfiguration architecture, would be to change the interface identifier portion of an address over time. Changing the interface identifier can make it more difficult to look at the IP addresses in independent transactions and identify which ones actually correspond to the same host, both in the case where the routing prefix portion of an address changes and when it does not.

Many hosts function as both clients and servers. In such cases, the host would need a name (e.g. a DNS domain name) for its use as a server. Whether the address stays fixed or changes has little privacy implication since the name remains constant and serves as a constant identifier. When acting as a client (e.g., initiating communication), however, such a host may want to vary the addresses it uses. In such environments, one may need multiple addresses: a stable address associated with the name, that is used to accept incoming connection requests from other hosts, and a temporary address used to shield the identity of the client when it initiates communication.

On the other hand, a host that functions only as a client may want to employ only temporary addresses for public communication.

To make it difficult to make educated guesses as to whether two different interface identifiers belong to the same host, the algorithm for generating alternate identifiers must include input that has an unpredictable component from the perspective of the outside entities that are collecting information.

3. Protocol Description

The following subsections define the procedures for the generation of IPv6 temporary addresses.

3.1. Design Guidelines

Temporary addresses observe the following properties:

1. Temporary addresses are typically employed for initiating outgoing sessions.
2. Temporary addresses are used for a short period of time (typically hours to days) and are subsequently deprecated. Deprecated addresses can continue to be used for established connections, but are not used to initiate new connections.
3. New temporary addresses are generated over time to replace temporary addresses that expire.
4. Temporary addresses must have a limited lifetime (limited "valid lifetime" and "preferred lifetime" from [RFC4862]). The lifetime of an address should be further reduced when privacy-meaningful events (such as a host attaching to a different network, or the regeneration of a new randomized MAC address) takes place. The lifetime of temporary addresses must be statistically different for different addresses, such that it is hard to predict or infer when a new temporary address is generated, or correlate a newly-generated address with an existing one.
5. By default, one address is generated for each prefix advertised by stateless address autoconfiguration. The resulting Interface Identifiers must be statistically different when addresses are configured for different prefixes or different network interfaces. This means that, given two addresses, it must be difficult for an outside entity to infer whether the addresses correspond to the same host or network interface.
6. It must be difficult for an outside entity to predict the Interface Identifiers that will be employed for temporary addresses, even with knowledge of the algorithm/method employed to generate them and/or knowledge of the Interface Identifiers previously employed for other temporary addresses. These Interface Identifiers must be semantically opaque [RFC7136] and must not follow any specific patterns.

3.2. Assumptions

The following algorithm assumes that for a given temporary address, an implementation can determine the prefix from which it was generated. When a temporary address is deprecated, a new temporary address is generated. The specific valid and preferred lifetimes for

the new address are dependent on the corresponding lifetime values set for the prefix from which it was generated.

Finally, this document assumes that when a host initiates outgoing communication, temporary addresses can be given preference over stable addresses (if available), when the device is configured to do so. [RFC6724] mandates implementations to provide a mechanism, which allows an application to configure its preference for temporary addresses over stable addresses. It also allows for an implementation to prefer temporary addresses by default, so that the connections initiated by the host can use temporary addresses without requiring application-specific enablement. This document also assumes that an API will exist that allows individual applications to indicate whether they prefer to use temporary or stable addresses and override the system defaults (see e.g. [RFC5014]).

3.3. Generation of Randomized Interface Identifiers

The following subsections specify example algorithms for generating temporary interface identifiers that follow the guidelines in Section 3.1 of this document. The algorithm specified in Section 3.3.1 benefits from a Pseudo-Random Number Generator (PRNG) available on the system. The algorithm specified in Section 3.3.2 allows for code reuse by hosts that implement [RFC7217].

3.3.1. Simple Randomized Interface Identifiers

One approach is to select a pseudorandom number of the appropriate length. A host employing this algorithm should generate IIDs as follows:

1. Obtain a random number from a pseudo-random number generator (PRNG) that can produce random numbers of at least as many bits as required for the Interface Identifier (please see the next step). [RFC4086] specifies randomness requirements for security.
2. The Interface Identifier is obtained by taking as many bits from the random number obtained in the previous step as necessary. See [RFC7136] for the necessary number of bits, that is, the length of the IID. See also [RFC7421] for a discussion of the privacy implications of the IID length. Note: there are no special bits in an Interface Identifier [RFC7136].
3. The resulting Interface Identifier MUST be compared against the reserved IPv6 Interface Identifiers [RFC5453] [IANA-RESERVED-IID] and against those Interface Identifiers already employed in an address of the same network interface and the same network prefix. In the event that an unacceptable identifier has been

generated, a new interface identifier should be generated, by repeating the algorithm from the first step.

3.3.2. Hash-based Generation of Randomized Interface Identifiers

The algorithm in [RFC7217] can be augmented for the generation of temporary addresses. The benefit of this would be that a host could employ a single algorithm for generating stable and temporary addresses, by employing appropriate parameters.

Hosts would employ the following algorithm for generating the temporary IID:

1. Compute a random identifier with the expression:

$RID = F(\text{Prefix}, \text{Net_Iface}, \text{Network_ID}, \text{Time}, \text{DAD_Counter}, \text{secret_key})$

Where:

RID:

Random Identifier

F():

A pseudorandom function (PRF) that MUST NOT be computable from the outside (without knowledge of the secret key). F() MUST also be difficult to reverse, such that it resists attempts to obtain the secret_key, even when given samples of the output of F() and knowledge or control of the other input parameters. F() SHOULD produce an output of at least as many bits as required for the Interface Identifier. F() could be the result of applying a cryptographic hash over an encoded version of the function parameters. While this document does not recommend a specific mechanism for encoding the function parameters (or a specific cryptographic hash function), a cryptographically robust construction will ensure that the mapping from parameters to the hash function input is an injective map, as might be attained by using fixed-width encodings and/or length-prefixing variable-length parameters. SHA-256 [FIPS-SHS] is one possible option for F(). Note: MD5 [RFC1321] is considered unacceptable for F() [RFC6151].

Prefix:

The prefix to be used for SLAAC, as learned from an ICMPv6 Router Advertisement message.

Net_Iface:

The MAC address corresponding to the underlying network interface card, in the case the link uses IEEE802 link-layer identifiers. Employing the MAC address for this parameter (over the other suggested options in RFC7217) means that the re-generation of a randomized MAC address will result in a different temporary address.

Network_ID:

Some network-specific data that identifies the subnet to which this interface is attached -- for example, the IEEE 802.11 Service Set Identifier (SSID) corresponding to the network to which this interface is associated. Additionally, "Simple Procedures for Detecting Network Attachment in IPv6" ("Simple DNA") [RFC6059] describes ideas that could be leveraged to generate a Network_ID parameter. This parameter SHOULD be employed if some form of "Network_ID" is available.

Time:

An implementation-dependent representation of time. One possible example is the representation in UNIX-like systems [OPEN-GROUP], that measure time in terms of the number of seconds elapsed since the Epoch (00:00:00 Coordinated Universal Time (UTC), 1 January 1970). The addition of the "Time" argument results in (statistically) different interface identifiers over time.

DAD_Counter:

A counter that is employed to resolve Duplicate Address Detection (DAD) conflicts.

secret_key:

A secret key that is not known by the attacker. The secret key SHOULD be of at least 128 bits. It MUST be initialized to a pseudo-random number (see [RFC4086] for randomness requirements for security) when the operating system is "bootstrapped". The secret_key MUST NOT be employed for any other purpose than the one discussed in this section. For example, implementations MUST NOT employ the same secret_key for the generation of stable addresses [RFC7217] and the generation of temporary addresses via this algorithm.

2. The Interface Identifier is finally obtained by taking as many bits from the RID value (computed in the previous step) as necessary, starting from the least significant bit. See [RFC7136] for the necessary number of bits, that is, the length of the IID. See also [RFC7421] for a discussion of the privacy implications of the IID length. Note: there are no special bits in an Interface Identifier [RFC7136].

3. The resulting Interface Identifier MUST be compared against the reserved IPv6 Interface Identifiers [RFC5453] [IANA-RESERVED-IID] and against those Interface Identifiers already employed in an address of the same network interface and the same network prefix. In the event that an unacceptable identifier has been generated, the value DAD_Counter should be incremented by 1, and the algorithm should be restarted from the first step.

3.4. Generating Temporary Addresses

[RFC4862] describes the steps for generating a link-local address when an interface becomes enabled as well as the steps for generating addresses for other scopes. This document extends [RFC4862] as follows. When processing a Router Advertisement with a Prefix Information option carrying a prefix for the purposes of address autoconfiguration (i.e., the A bit is set), the host MUST perform the following steps:

1. Process the Prefix Information Option as defined in [RFC4862], adjusting the lifetimes of existing temporary addresses, with the overall constraint that no temporary addresses should ever remain "valid" or "preferred" for a time longer than (TEMP_VALID_LIFETIME) or (TEMP_PREFERRED_LIFETIME - DESYNC_FACTOR) respectively. The configuration variables TEMP_VALID_LIFETIME and TEMP_PREFERRED_LIFETIME correspond to maximum target lifetimes for temporary addresses.
2. One way an implementation can satisfy the above constraints is to associate with each temporary address a creation time (called CREATION_TIME) that indicates the time at which the address was created. When updating the preferred lifetime of an existing temporary address, it would be set to expire at whichever time is earlier: the time indicated by the received lifetime or (CREATION_TIME + TEMP_PREFERRED_LIFETIME - DESYNC_FACTOR). A similar approach can be used with the valid lifetime.
3. If the host has not configured any temporary address for the corresponding prefix, the host SHOULD create a new temporary address for such prefix.

Note:

For example, a host might implement prefix-specific policies such as not configuring temporary addresses for the Unique Local IPv6 Unicast Addresses (ULA) [RFC4193] prefix.

4. When creating a temporary address, the DESYNC_FACTOR MUST be computed for this prefix, and the lifetime values MUST be derived from the corresponding prefix as follows:

- * Its Valid Lifetime is the lower of the Valid Lifetime of the prefix and TEMP_VALID_LIFETIME.
 - * Its Preferred Lifetime is the lower of the Preferred Lifetime of the prefix and TEMP_PREFERRED_LIFETIME - DESYNC_FACTOR.
5. A temporary address is created only if this calculated Preferred Lifetime is greater than REGEN_ADVANCE time units. In particular, an implementation MUST NOT create a temporary address with a zero Preferred Lifetime.
 6. New temporary addresses MUST be created by appending a randomized interface identifier to the prefix that was received. Section 3.3 of this document specifies some sample algorithms for generating the randomized interface identifier.
 7. The host MUST perform duplicate address detection (DAD) on the generated temporary address. If DAD indicates the address is already in use, the host MUST generate a new randomized interface identifier, and repeat the previous steps as appropriate up to TEMP_IDGEN_RETRIES times. If after TEMP_IDGEN_RETRIES consecutive attempts no non-unique address was generated, the host MUST log a system error and SHOULD NOT attempt to generate a temporary address for the given prefix for the duration of the host's attachment to the network via this interface. This allows hosts to recover from occasional DAD failures, or otherwise log the recurrent address collisions.

3.5. Expiration of Temporary Addresses

When a temporary address becomes deprecated, a new one MUST be generated. This is done by repeating the actions described in Section 3.4, starting at step 4). Note that, except for the transient period when a temporary address is being regenerated, in normal operation at most one temporary address per prefix should be in a non-deprecated state at any given time on a given interface. Note that if a temporary address becomes deprecated as result of processing a Prefix Information Option with a zero Preferred Lifetime, then a new temporary address MUST NOT be generated. To ensure that a preferred temporary address is always available, a new temporary address SHOULD be regenerated slightly before its predecessor is deprecated. This is to allow sufficient time to avoid race conditions in the case where generating a new temporary address is not instantaneous, such as when duplicate address detection must be run. The host SHOULD start the address regeneration process REGEN_ADVANCE time units before a temporary address would actually be deprecated.

As an optional optimization, an implementation MAY remove a deprecated temporary address that is not in use by applications or upper layers as detailed in Section 6.

3.6. Regeneration of Temporary Addresses

The frequency at which temporary addresses change depends on how a device is being used (e.g., how frequently it initiates new communication) and the concerns of the end user. The most egregious privacy concerns appear to involve addresses used for long periods of time (weeks to months to years). The more frequently an address changes, the less feasible collecting or coordinating information keyed on interface identifiers becomes. Moreover, the cost of collecting information and attempting to correlate it based on interface identifiers will only be justified if enough addresses contain non-changing identifiers to make it worthwhile. Thus, having large numbers of clients change their address on a daily or weekly basis is likely to be sufficient to alleviate most privacy concerns.

There are also client costs associated with having a large number of addresses associated with a host (e.g., in doing address lookups, the need to join many multicast groups, etc.). Thus, changing addresses frequently (e.g., every few minutes) may have performance implications.

Hosts following this specification SHOULD generate new temporary addresses on a periodic basis. This can be achieved by generating a new temporary address at least once every $(\text{TEMP_PREFERRED_LIFETIME} - \text{REGEN_ADVANCE} - \text{DESYNC_FACTOR})$ time units. As described above, generating a new temporary address REGEN_ADVANCE time units before a temporary address becomes deprecated produces addresses with a preferred lifetime no larger than $\text{TEMP_PREFERRED_LIFETIME}$. The value DESYNC_FACTOR is a random value computed for a prefix when a temporary address is generated, that ensures that clients do not generate new addresses with a fixed frequency, and that clients do not synchronize with each other and generate new addresses at exactly the same time. When the preferred lifetime expires, a new temporary address MUST be generated using the algorithm specified in Section 3.4.

Because the precise frequency at which it is appropriate to generate new addresses varies from one environment to another, implementations SHOULD provide end users with the ability to change the frequency at which addresses are regenerated. The default value is given in $\text{TEMP_PREFERRED_LIFETIME}$ and is one day. In addition, the exact time at which to invalidate a temporary address depends on how applications are used by end users. Thus, the suggested default value of two days ($\text{TEMP_VALID_LIFETIME}$) may not be appropriate in all

environments. Implementations SHOULD provide end users with the ability to override both of these default values.

Finally, when an interface connects to a new (different) link, existing temporary addresses for the corresponding interface MUST be eliminated, and new temporary addresses MUST be generated immediately for use on the new link. If a device moves from one link to another, generating new temporary addresses ensures that the device uses different randomized interface identifiers for the temporary addresses associated with the two links, making it more difficult to correlate addresses from the two different links as being from the same hosts. The host MAY follow any process available to it, to determine that the link change has occurred. One such process is described by Simple DNA [RFC6059]. Detecting link changes would prevent link down/up events from causing temporary addresses to be (unnecessarily) regenerated.

3.7. Implementation Considerations

Devices implementing this specification MUST provide a way for the end user to explicitly enable or disable the use of temporary addresses. In addition, a site might wish to disable the use of temporary addresses in order to simplify network debugging and operations. Consequently, implementations SHOULD provide a way for trusted system administrators to enable or disable the use of temporary addresses.

Additionally, sites might wish to selectively enable or disable the use of temporary addresses for some prefixes. For example, a site might wish to disable temporary address generation for "Unique local" [RFC4193] prefixes while still generating temporary addresses for all other global prefixes. Another site might wish to enable temporary address generation only for the prefixes 2001:db8:1::/48 and 2001:db8:2::/48 while disabling it for all other prefixes. To support this behavior, implementations SHOULD provide a way to enable and disable generation of temporary addresses for specific prefix subranges. This per-prefix setting SHOULD override the global settings on the host with respect to the specified prefix subranges. Note that the per-prefix setting can be applied at any granularity, and not necessarily on a per subnet basis.

3.8. Defined Constants and Configuration Variables

Constants and configuration variables defined in this document include:

TEMP_VALID_LIFETIME -- Default value: 2 days. Users should be able to override the default value.

TEMP_PREFERRED_LIFETIME -- Default value: 1 day. Users should be able to override the default value.

Note:

The TEMP_PREFERRED_LIFETIME value MUST be less than the TEMP_VALID_LIFETIME value, to avoid the pathological case where an address is employed for new communications, but becomes invalid in less than 1 second, disrupting those communications

REGEN_ADVANCE -- $2 + (\text{TEMP_IDGEN_RETRIES} * \text{DupAddrDetectTransmits} * \text{RetransTimer} / 1000)$

Notes:

This parameter is specified as a function of other protocol parameters, to account for the time possibly spent in Duplicate Address Detection (DAD) in the worst-case scenario of TEMP_IDGEN_RETRIES. This prevents the pathological case where the generation of a new temporary address is not started with enough anticipation such that a new preferred address is generated before the currently-preferred temporary address becomes deprecated.

RetransTimer is specified in [RFC4861], while DupAddrDetectTransmits is specified in [RFC4862]. Since RetransTimer is specified in units of milliseconds, this expression employs the constant "1000" such that REGEN_ADVANCE is expressed in seconds.

MAX_DESYNC_FACTOR -- $0.4 * \text{TEMP_PREFERRED_LIFETIME}$. Upper bound on DESYNC_FACTOR.

Note:

Setting MAX_DESYNC_FACTOR to $0.4 * \text{TEMP_PREFERRED_LIFETIME}$ results in addresses that have statistically different lifetimes, and a maximum of 3 concurrent temporary addresses when the default parameters specified in this section are employed.

DESYNC_FACTOR -- A random value within the range 0 - MAX_DESYNC_FACTOR. It is computed for a prefix each time a temporary address is generated, and must be smaller than $(\text{TEMP_PREFERRED_LIFETIME} - \text{REGEN_ADVANCE})$.

TEMP_IDGEN_RETRIES -- Default value: 3

4. Implications of Changing Interface Identifiers

The desire to protect individual privacy can conflict with the desire to effectively maintain and debug a network. Having clients use addresses that change over time will make it more difficult to track

down and isolate operational problems. For example, when looking at packet traces, it could become more difficult to determine whether one is seeing behavior caused by a single errant host, or by a number of them.

Network deployments are currently recommended to provide multiple IPv6 addresses from each prefix to general-purpose hosts [RFC7934]. However, in some scenarios, use of a large number of IPv6 addresses may have negative implications on network devices that need to maintain entries for each IPv6 address in some data structures (e.g., [RFC7039]). For example, concurrent active use of multiple IPv6 addresses will increase neighbor discovery traffic if Neighbor Caches in network devices are not large enough to store all addresses on the link. This can impact performance and energy efficiency on networks on which multicast is expensive (e.g. [I-D.ietf-mboned-ieee802-mcast-problems]). Additionally, some network security devices might incorrectly infer IPv6 address forging if temporary addresses are regenerated at a high rate.

The use of temporary addresses may cause unexpected difficulties with some applications. For example, some servers refuse to accept communications from clients for which they cannot map the IP address into a DNS name. That is, they perform a DNS PTR query to determine the DNS name, and may then also perform an AAAA query on the returned name to verify that the returned DNS name maps back into the address being used. Consequently, clients not properly registered in the DNS may be unable to access some services. However, a host's DNS name (if non-changing) would serve as a constant identifier. The wide deployment of the extension described in this document could challenge the practice of inverse-DNS-based "validation", which has little validity, though it is widely implemented. In order to meet server challenges, hosts could register temporary addresses in the DNS using random names (for example, a string version of the random address itself), albeit at the expense of increased complexity.

In addition, some applications may not behave robustly if temporary addresses are used and an address expires before the application has terminated, or if it opens multiple sessions, but expects them to all use the same addresses.

[RFC4941] employed a randomized temporary Interface Identifier for generating a set of temporary addresses, such that temporary addresses configured at a given time for multiple SLAAC prefixes would employ the same Interface Identifier. Sharing the same IID among multiple address allowed host to join only one solicited-node multicast group per temporary address set.

This document requires that the Interface Identifiers of all temporary addresses on a host are statistically different from each other. This means that when a network employs multiple prefixes, each temporary address of a set will result in a different solicited-node multicast address, and thus the number of multicast groups that a host must join becomes a function of the number of SLAAC prefixes employed for generating temporary addresses.

Thus, a network that employs multiple prefixes may require hosts to join more multicast groups than for an RFC4941 implementation. If the number of multicast groups were large enough, a node might need to resort to setting the network interface card to promiscuous mode. This could cause the node to process more packets than strictly necessary, and might have a negative impact on battery-life, and on system performance in general.

We note that since this document reduces the default TEMP_VALID_LIFETIME from 7 days (in [RFC4941]) to 2 days, the number of concurrent temporary addresses per SLAAC prefix will be smaller than for RFC4941 implementations, and thus the number of multicast groups for a network that employs, say, between 1 and three prefixes will be similar than of RFC4941 implementations.

Implementations concerned with the maximum number of multicast groups that would be required to join as a result of configured addresses, or the overall number of configured addresses, should consider enforcing implementation-specific limits on e.g. the maximum number of configured addresses, the maximum number of SLAAC prefixes that are employed for auto-configuration, and/or the maximum ratio for TEMP_VALID_LIFETIME/TEMP_PREFERRED_LIFETIME (that ultimately controls the approximate number of concurrent temporary addresses per SLAAC prefix). Many of these configuration limits are readily available in SLAAC and RFC4941 implementations. We note that these configurable limits are meant to prevent pathological behaviors (as opposed to simply limiting the usage of IPv6 addresses), since IPv6 implementations are expected to leverage the usage of multiple addresses [RFC7934].

5. Significant Changes from RFC4941

This section summarizes the substantive changes in this document relative to RFC 4941.

Broadly speaking, this document introduces the following changes:

- o Addresses a number of flaws in the algorithm for generating temporary addresses: The aforementioned flaws include the use of MD5 for computing the temporary IIDs, and reusing the same IID for

multiple prefixes (see [RAID2015] and [RFC7721] for further details).

- o Allows hosts to employ only temporary addresses: [RFC4941] assumed that temporary addresses were configured in addition to stable addresses. This document does not imply or require the configuration of stable addresses, and thus implementations can now configure both stable and temporary addresses, or temporary addresses only.
- o Removes the recommendation that temporary addresses be disabled by default:
This is in line with BCP188 ([RFC7258]), and also with BCP204 ([RFC7934]).
- o Reduces the default maximum Valid Lifetime for temporary addresses: The default Valid Lifetime for temporary addresses has been reduced from 1 week to 2 days, decreasing the typical number of concurrent temporary addresses from 7 to 3. This reduces the possible stress on network elements (see Section 4 for further details).
- o DESYNC_FACTOR is computed on a per-prefix basis each time a temporary address is generated, such that each temporary address has a statistically different preferred lifetime, and that temporary addresses are not generated at a constant frequency.
- o Changes the requirement to not try to regenerate temporary addresses upon DAD failures from "MUST NOT" to "SHOULD NOT".
- o The discussion about the security and privacy implications of different address generation techniques has been replaced with references to recent work in this area ([RFC7707], [RFC7721], and [RFC7217]).
- o Addresses all errata submitted for [RFC4941].

6. Future Work

An implementation might want to keep track of which addresses are being used by upper layers so as to be able to remove a deprecated temporary address from internal data structures once no upper layer protocols are using it (but not before). This is in contrast to current approaches where addresses are removed from an interface when they become invalid [RFC4862], independent of whether or not upper layer protocols are still using them. For TCP connections, such information is available in control blocks. For UDP-based applications, it may be the case that only the applications have

knowledge about what addresses are actually in use. Consequently, an implementation generally will need to use heuristics in deciding when an address is no longer in use.

7. Implementation Status

[The RFC-Editor should remove this section before publishing this document as an RFC]

The following are known implementations of this document:

- o FreeBSD kernel: There is a FreeBSD kernel implementation of this document, albeit not yet committed. The implementation has been done in April 2020 by Fernando Gont <fgont@si6networks.com>. The corresponding patch can be found at:
<<https://www.gont.com.ar/code/fgont-patch-freebsd-rfc4941bis.txt>>
- o Linux kernel: A Linux kernel implementation of this document has been committed to the net-next tree. The implementation has been produced in April 2020 by Fernando Gont <fgont@si6networks.com>. The corresponding patch can be found at:
<<https://patchwork.ozlabs.org/project/netdev/patch/20200501035147.GA1587@archlinux-current.localdomain/>>
- o slaacd(8): slaacd(8) has traditionally used different randomized interface identifiers for each prefix, and it has recently reduced the Valid Lifetime of temporary addresses as specified in Section 3.8, thus fully implementing this document. The implementation has been done by Florian Obser <florian@openbsd.org>, with the update to the temporary address Valid Lifetime applied in March 2020. The implementation can be found at: <<https://github.com/openbsd/src/tree/master/sbin/slaacd>>

8. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

9. Security Considerations

If a very small number of hosts (say, only one) use a given prefix for extended periods of time, just changing the interface identifier part of the address may not be sufficient to mitigate address-based network activity correlation, since the prefix acts as a constant identifier. The procedures described in this document are most effective when the prefix is reasonably non static or is used by a fairly large number of hosts. Additionally, if a temporary address

is used in a session where the user authenticates, any notion of "privacy" for that address is compromised for the part(ies) that receive the authentication information.

While this document discusses ways to limit the lifetime of Interface Identifiers to reduce the ability of attackers to perform address-based network activity correlation, the method described is believed to be ineffective against sophisticated forms of traffic analysis. To increase effectiveness, one may need to consider the use of more advanced techniques, such as Onion Routing [ONION].

Ingress filtering has been and is being deployed as a means of preventing the use of spoofed source addresses in Distributed Denial of Service (DDoS) attacks. In a network with a large number of hosts, new temporary addresses are created at a fairly high rate. This might make it difficult for ingress filtering mechanisms to distinguish between legitimately changing temporary addresses and spoofed source addresses, which are "in-prefix" (using a topologically correct prefix and non-existent interface ID). This can be addressed by using access control mechanisms on a per-address basis on the network egress point, though as noted in Section 4 there are corresponding costs for doing so.

10. Acknowledgments

The authors would like to thank (in alphabetical order) Fred Baker, Brian Carpenter, Tim Chown, Lorenzo Colitti, Roman Danyliw, David Farmer, Tom Herbert, Bob Hinden, Christian Huitema, Benjamin Kaduk, Erik Kline, Gyan Mishra, Dave Plonka, Alvaro Retana, Michael Richardson, Mark Smith, Dave Thaler, Pascal Thubert, Ole Troan, Johanna Ullrich, Eric Vyncke, and Timothy Winters, for providing valuable comments on earlier versions of this document.

This document incorporates errata submitted for [RFC4941] by Jiri Bohac and Alfred Hoenes.

This document is based on [RFC4941] (a revision of RFC3041). Suresh Krishnan was the sole author of RFC4941. He would like to acknowledge the contributions of the IPv6 working group and, in particular, Jari Arkko, Pekka Nikander, Pekka Savola, Francis Dupont, Brian Haberman, Tatuya Jinmei, and Margaret Wasserman for their detailed comments.

Rich Draves and Thomas Narten were the authors of RFC 3041. They would like to acknowledge the contributions of the IPv6 working group and, in particular, Ran Atkinson, Matt Crawford, Steve Deering, Allison Mankin, and Peter Bieringer.

11. References

11.1. 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>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [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>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC5453] Krishnan, S., "Reserved IPv6 Interface Identifiers", RFC 5453, DOI 10.17487/RFC5453, February 2009, <<https://www.rfc-editor.org/info/rfc5453>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136, February 2014, <<https://www.rfc-editor.org/info/rfc7136>>.
- [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>>.

11.2. Informative References

[FIPS-SHS]

NIST, "Secure Hash Standard (SHS)", FIPS Publication 180-4, August 2015, <<https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>>.

[I-D.ietf-mboned-ieee802-mcast-problems]

Perkins, C., McBride, M., Stanley, D., Kumari, W., and J. Zuniga, "Multicast Considerations over IEEE 802 Wireless Media", draft-ietf-mboned-ieee802-mcast-problems-12 (work in progress), October 2020.

[IANA-RESERVED-IID]

IANA, "Reserved IPv6 Interface Identifiers", <<http://www.iana.org/assignments/ipv6-interface-ids>>.

[ONION]

Reed, MGR., Syverson, PFS., and DMG. Goldschlag, "Proxies for Anonymous Routing", Proceedings of the 12th Annual Computer Security Applications Conference, San Diego, CA, December 1996.

[OPEN-GROUP]

The Open Group, "The Open Group Base Specifications Issue 7 / IEEE Std 1003.1-2008, 2016 Edition", Section 4.16 Seconds Since the Epoch, 2016, <<http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/contents.html>>.

[RAID2015]

Ullrich, J. and E. Weippl, "Privacy is Not an Option: Attacking the IPv6 Privacy Extension", International Symposium on Recent Advances in Intrusion Detection (RAID), 2015, <<https://www.sba-research.org/wp-content/uploads/publications/Ullrich2015Privacy.pdf>>.

[RFC1321]

Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <<https://www.rfc-editor.org/info/rfc1321>>.

[RFC4941]

Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.

- [RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, DOI 10.17487/RFC5014, September 2007, <<https://www.rfc-editor.org/info/rfc5014>>.
- [RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachment in IPv6", RFC 6059, DOI 10.17487/RFC6059, November 2010, <<https://www.rfc-editor.org/info/rfc6059>>.
- [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <<https://www.rfc-editor.org/info/rfc6151>>.
- [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <<https://www.rfc-editor.org/info/rfc6265>>.
- [RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <<https://www.rfc-editor.org/info/rfc7039>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<https://www.rfc-editor.org/info/rfc7217>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <<https://www.rfc-editor.org/info/rfc7421>>.
- [RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016, <<https://www.rfc-editor.org/info/rfc7707>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<https://www.rfc-editor.org/info/rfc7721>>.

- [RFC7934] Colitti, L., Cerf, V., Cheshire, S., and D. Schinazi,
"Host Address Availability Recommendations", BCP 204,
RFC 7934, DOI 10.17487/RFC7934, July 2016,
<<https://www.rfc-editor.org/info/rfc7934>>.
- [RFC8190] Bonica, R., Cotton, M., Haberman, B., and L. Vegoda,
"Updates to the Special-Purpose IP Address Registries",
BCP 153, RFC 8190, DOI 10.17487/RFC8190, June 2017,
<<https://www.rfc-editor.org/info/rfc8190>>.

Authors' Addresses

Fernando Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: <https://www.si6networks.com>

Suresh Krishnan
Kaloom

Email: suresh@kaloom.com

Thomas Narten

Email: narten@cs.duke.edu

Richard Draves
Microsoft Research
One Microsoft Way
Redmond, WA
USA

Email: richdr@microsoft.com

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.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. Segment Routing Header	4
2.1. SRH TLVs	6
2.1.1. Padding TLVs	8
2.1.2. HMAC TLV	9
3. SR Nodes	12
3.1. Source SR Node	12
3.2. Transit Node	12
3.3. SR Segment Endpoint Node	12
4. Packet Processing	13
4.1. Source SR Node	13
4.1.1. Reduced SRH	13
4.2. Transit Node	14
4.3. SR Segment Endpoint Node	14
4.3.1. FIB Entry Is Locally Instantiated SRv6 SID	14
4.3.2. FIB Entry Is A Local Interface	16
4.3.3. FIB Entry Is A Non-Local Route	17
4.3.4. FIB Entry Is A No Match	17
5. Intra SR Domain Deployment Model	17
5.1. Securing the SR Domain	17
5.2. SR Domain as A Single System with Delegation Among Components	18
5.3. MTU Considerations	19
5.4. ICMP Error Processing	19
5.5. Load Balancing and ECMP	19
5.6. Other Deployments	20
6. Illustrations	20
6.1. Abstract Representation of an SRH	20
6.2. Example Topology	21
6.3. Source SR Node	22
6.3.1. Intra SR Domain Packet	22
6.3.2. Inter SR Domain Packet - Transit	22
6.3.3. Inter SR Domain Packet - Internal to External	23
6.4. Transit Node	23
6.5. SR Segment Endpoint Node	23
6.6. Delegation of Function with HMAC Verification	23
6.6.1. SID List Verification	24

7.	Security Considerations	24
7.1.	Source Routing Attacks	25
7.2.	Service Theft	25
7.3.	Topology Disclosure	25
7.4.	ICMP Generation	26
7.5.	Applicability of AH	26
8.	IANA Considerations	26
8.1.	Segment Routing Header Flags Registry	27
8.2.	Segment Routing Header TLVs Registry	27
9.	Implementation Status	27
9.1.	Linux	28
9.2.	Cisco Systems	28
9.3.	FD.io	28
9.4.	Barefoot	28
9.5.	Juniper	28
9.6.	Huawei	29
10.	Contributors	29
11.	Acknowledgements	29
12.	References	29
12.1.	Normative References	29
12.2.	Informative References	31
	Authors' Addresses	32

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].

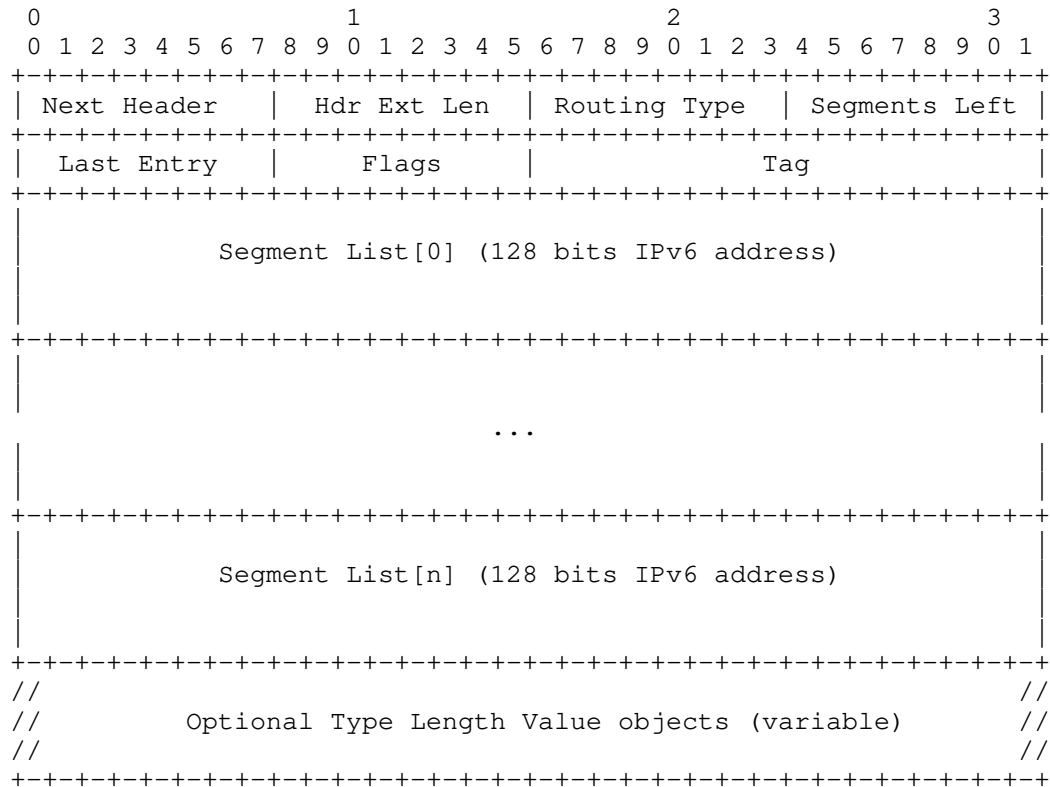
1.1. Requirements Language

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:

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

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

```

  0 1 2 3 4 5 6 7
+-----+
|U U U U U U U U|
+-----+

```

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

- o 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.
- o 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.
- o 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.

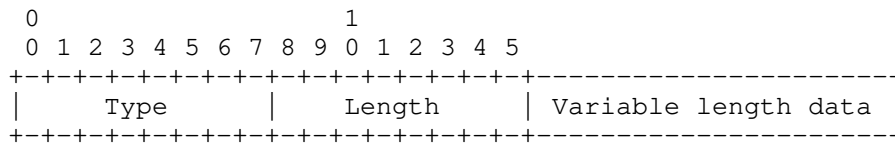
TLVs are present when the Hdr Ext Len is greater than (Last Entry+1)*2.

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:

- o 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.
- o Limit the length in PadN to 5.
- o Limit the maximum number of non-Pad TLVs to be processed.
- o 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.

2.1.1.1. 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

2.1.1.1.1. PAD1

Alignment requirement: none

```

0 1 2 3 4 5 6 7
+---+---+---+---+
|           Type           |
+---+---+---+---+

```

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.

2.1.1.1.2. PADN

Alignment requirement: none

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Type           |           Length           |           Padding (variable)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
//                               Padding (variable)                               //
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

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

Length: 0 to 5

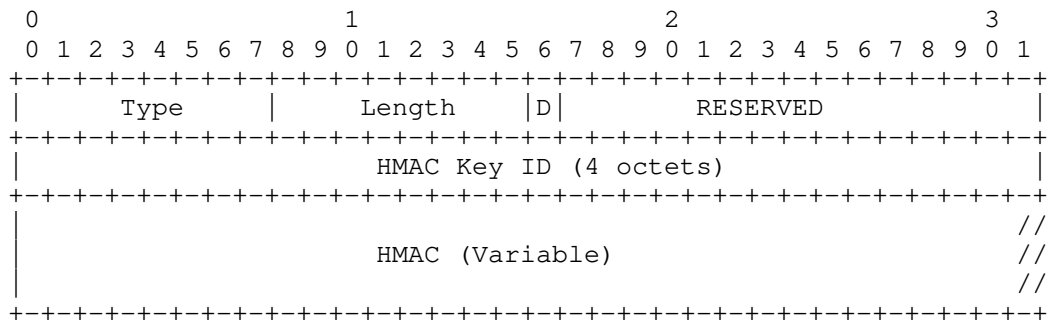
Padding: Length octets of padding. Padding bits have no semantic. They MUST be set to 0 on transmission and ignored on receipt.

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

2.1.1.2. HMAC TLV

Alignment requirement: 8n

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



where:

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

- o HMAC D bit is 1 and Segments Left is greater than Last Entry.
- o 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:

- o key: the pre-shared key identified by HMAC Key ID
- o HMAC algorithm: identified by the HMAC Key ID
- o 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:

- o 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:

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

- o 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 A_k , its SRv6 SID is represented as S_k .

IPv6 headers are represented as the tuple of (source, destination). For example, a packet with source address A_1 and destination address A_2 is represented as (A_1, A_2) . The payload of the packet is omitted.

An SR Policy is a list of segments. A list of segments is represented as $\langle S_1, S_2, S_3 \rangle$ where S_1 is the first SID to visit, S_2 is the second SID to visit and S_3 is the last SID to visit.

$(SA, DA) (S_3, S_2, S_1; SL)$ represents an IPv6 packet with:

- o Source Address is SA, Destination Addresses is DA, and next-header is SRH.
- o SRH with SID list $\langle S_1, S_2, S_3 \rangle$ with SegmentsLeft = SL.

- o 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:

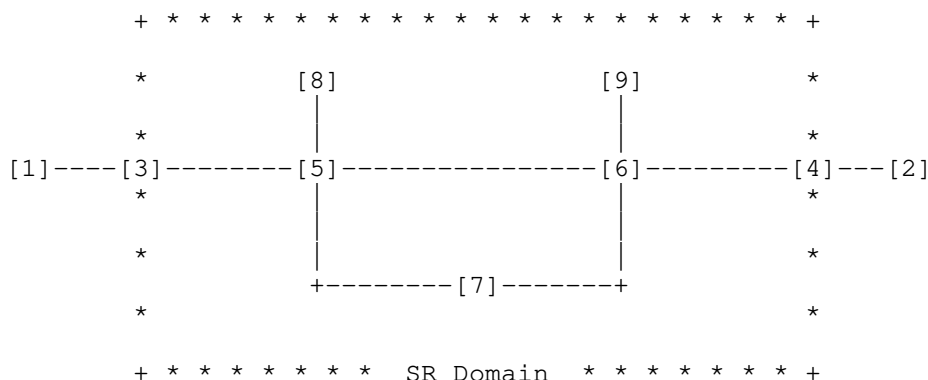


Figure 3

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

- o 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

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

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	5
3. Operational Considerations	5
4. IPv6 Destination Options	6
4.1. The Truncation Eligible Option	6
4.2. The Truncated Packet Option	7
5. Reference Topology	8
6. Truncation Procedures	8
7. Additional Truncation Considerations	10
8. Backwards Compatibility	10
9. Checksum Considerations	11
10. Invalid Packet Types	11
11. Network Considerations	12
12. Encapsulating Security Payload Considerations	12
13. Extension Header Considerations	13
14. Security Considerations	13
15. IANA Considerations	14
16. Acknowledgements	14
17. References	14
17.1. Normative References	14
17.2. Informative References	15
Authors' Addresses	15

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:

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

- o Mark the packet as being eligible for truncation.
- o Forward the packet towards its destination.

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

- o Detect that the packet is eligible for truncation.
- o 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.
- o Truncate the packet.
- o Mark the packet as being truncated.
- o Update the packet's upper-layer checksum (if possible).
- o Forward the packet towards its destination.

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

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

- o The network can deliver ICMP messages from the destination node to the source node.
- o 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:

- o The destination node has a viable route to the source node, but the intermediate node does not.
- o 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.
- o 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:

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

- o 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:

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

- o MAY include a single instance of the Truncation Eligible option.
- o SHOULD NOT include multiple instances of the Truncation Eligible option.
- o 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:

- o Option Type - Truncated Packet option. Value TBD by IANA. See Notes below.
- o 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:

- o MAY include a single instance of the Truncated Packet option.
- o SHOULD NOT include multiple instances of the Truncated Packet option.
- o 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

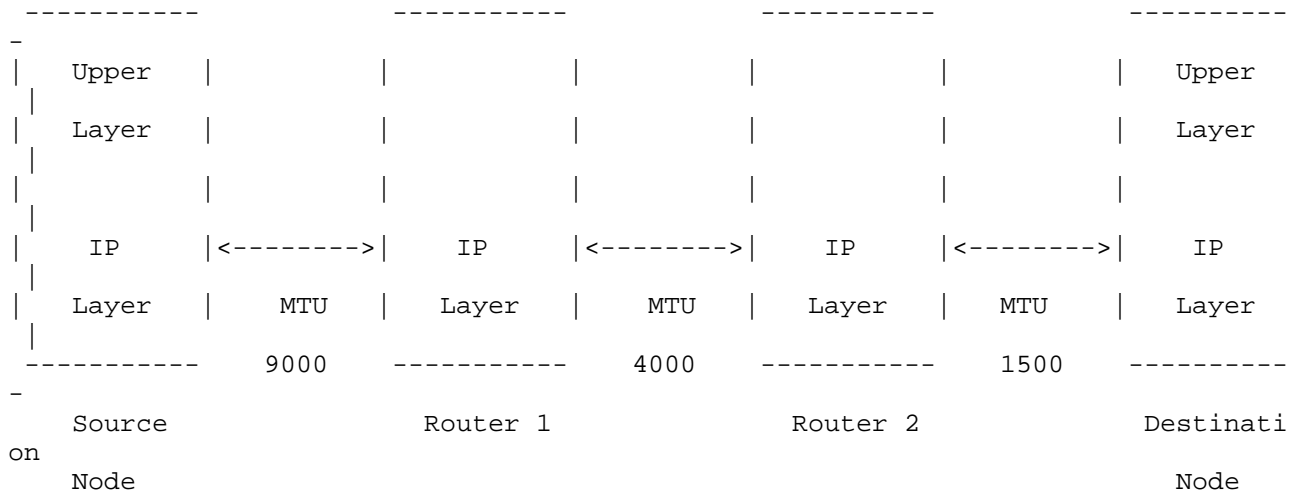


Figure 1: Reference Topology

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:

- o 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).
- o Truncates the packet, so that its total length equals the MTU of the next link in the path.
- o Updates the IPv6 Payload Length.
- o Overwrites all instances of the Truncation Eligible option with a Truncated Packet option.
- o Updates the upper-layer checksum (if possible)
- o 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 an 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 an 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:

- o Packets that contain the Fragment header and the Truncation Eligible option.
- o Packets that contain the Fragment header and the Packet Truncated option.
- o Packets that contain the Truncation Eligible option and the Packet Truncated option.
- o Packets that specify an Option Data Length greater than 0 in the Truncation Eligible option.

- o Packets that specify an Option Data Length greater than 0 in the Truncated Packet option.
- o 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:

- o Discard the packet
- o 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:

- o To convey packets that contain destination options from the source node to the destination node.
- o 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:

- o An Encapsulating Security Payload (ESP) [RFC4303] header.

- o 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 that 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:

- o The Hop-by-hop Options header.
- o 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:

- o Intermediate nodes, on an exception basis (i.e, when the packet cannot be forwarded due to MTU issues).
- o 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>).

- o Truncation Eligible ("act-bits" are "00". "chg-bit" can be either 0 or 1.)
- o 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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 5, 2021

F. Templin, Ed.
Boeing Research & Technology
January 1, 2021

A Unified Stateful/Stateless Configuration Service for IPv6
draft-templin-6man-dhcpv6-ndopt-11

Abstract

IPv6 Neighbor Discovery (IPv6ND) specifies a control message set for nodes to discover neighbors, routers, prefixes and other services on the link. It also supports a manner of Stateless Address AutoConfiguration (SLAAC), while the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) specifies a separate stateful service. This document presents IPv6ND extensions for providing a unified stateful/stateless configuration service.

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 July 5, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. DHCPv6 Options in IPv6 ND Messages	4
2.1. The DHCPv6 Option	4
2.2. DHCPv6 Option Usage	5
2.3. Stateful Provisioning Requirements	6
2.4. Implementation Considerations	7
3. PIO Options in RS Messages	7
3.1. The PIO Option in RS Messages	7
3.2. PIO Option Usage	7
3.3. Stateful Provisioning Requirements	8
3.4. Implementation Considerations	8
4. Embedded Prefix Assertion	9
4.1. Embedded Prefix Assertion	9
4.2. Embedded Prefix Usage	9
4.3. Stateful Provisioning Requirements	9
4.4. Implementation Considerations	10
5. Out-of-Band Network Login Messaging	10
5.1. Out-of-Band Network Login	10
5.2. Out-of-Band Network Login Usage	10
5.3. Stateful Provisioning Requirements	11
5.4. Implementation Considerations	11
6. Implementation Status	11
7. IANA Considerations	11
8. Security Considerations	11
9. Acknowledgements	12
10. References	12
10.1. Normative References	12
10.2. Informative References	12
Appendix A. Change Log	13
Author's Address	14

1. Introduction

IPv6 Neighbor Discovery (IPv6ND) [RFC4861] specifies a control message set for nodes to discover neighbors, routers, prefixes and other services on the link. It also supports a manner of Stateless Address AutoConfiguration (SLAAC). The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) specifies a separate service for delegation of prefixes, addresses and any other stateful information [RFC8415]. This document presents IPv6ND extensions for providing a unified stateful/stateless configuration service.

If the network can provide such a unified service, multi-message procedures can be condensed into a single and concise message exchange. This would ease network management as well as simplify host and router operations. It would further accommodate both stateless and stateful services in a way that combines the best aspects of both. The operating model is based on harnessing the IPv6 ND Router Solicitation (RS) / Router Advertisement (RA) functions to provide all configuration information in a single message exchange.

When a node first comes onto a link, it sends an RS to elicit an RA from one or more routers for the link. If the node also needs to acquire stateful information it then sends a DHCPv6 Solicit message to elicit a Reply message from a DHCPv6 server. This two round-trip message exchange can add delay as well as waste critical link bandwidth on low-end links (e.g., 6LoWPAN, satellite communications, aeronautical wireless, etc.). While it is possible to start both round trip exchanges at the same time, this would still result in twice as many channel access transactions as necessary. Moreover, the multicast nature of these messages could disturb other nodes on the link, e.g., resulting in an unnecessary wakeup from sleep mode.

This document proposes methods for combining all stateless and stateful configuration operations into a single, unified exchange based on IPv6ND messaging extensions. It notes that stateful exchanges should include:

- o an explicit request for stateful information
- o the identity of the requesting node
- o a transaction identification that the requesting node can use to match replies with their corresponding requests
- o any security parameters necessary for the requesting node to establish its authorization to receive stateful information

The first method is through definition of a new IPv6ND option called the "DHCPv6 Option" that combines the IPv6ND router discovery and DHCPv6 stateful processes into a single message exchange. Nodes include the DHCPv6 option in RS messages to solicit an RA message with a DHCPv6 option in return. This allows the IPv6ND and DHCPv6 functions to work together to supply the client with all needed configuration information in a minimum number of messages.

The second method proposes the inclusion of Prefix Information Options (PIOs) in RS messages for the purpose of soliciting stateful information. [I-D.naveen-slaac-prefix-management] discusses the

maintenance and management functions required for supporting the operation.

The third method entails the encoding of a prefix in the IPv6 link-local source address of the RS message. If the node is pre-configured with the prefix that it will solicit from the network, and if the network has a way of accepting the node's prefix assertion without the threat of spoofing, the network can then delegate the prefix and establish the necessary routing information.

The fourth method uses out-of-band messaging for the node to request stateful information outside of the scope of IPv6ND messaging. The out-of-band messaging could entail some sort of network login process (e.g., through Layer-2 (L2) messaging, etc.).

The following sections present considerations for nodes that employ these approaches.

2. DHCPv6 Options in IPv6 ND Messages

The first method entails the inclusion of DHCPv6 messages within IPv6ND RS and RA messages, as discussed in the following sections.

2.1. The DHCPv6 Option

The DHCPv6 option is a new IPv6ND option that simply embeds a standard DHCPv6 message per section 6 of [RFC8415], beginning with the 'msg-type' followed by the 'transaction-id' and all DHCPv6 'options'. The format of the option is as follows:

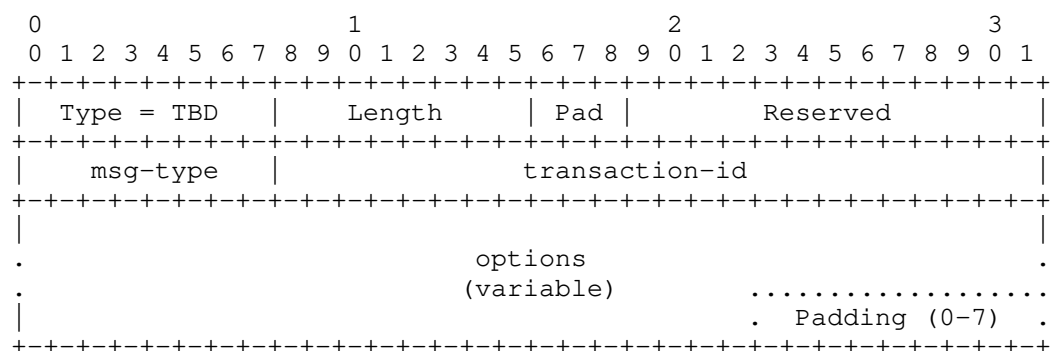


Figure 1: IPv6 ND DHCPv6 Option Format

In this format, 'Type' and 'Length' are exactly as defined in Section 4.6 of [RFC4861], 'Pad' is a 3-bit integer that encodes the padding length, 'Reserved' is included for alignment and future use,

and the rest of the option is formatted as specified in Section 6 of [RFC8415] except with trailing null padding added as necessary for 8 octet alignment. The length of the full DHCPv6 message is determined by $((\text{'Length'} * 8) - 4) - \text{'Pad'}$, for a maximum message length of 2036 octets.

The 'Reserved' field MUST be set to 0 on transmission and ignored on reception. Future specifications MAY define new uses for these bits.

2.2. DHCPv6 Option Usage

When a node first comes onto the link, it creates an RS message containing a DHCPv6 option that embeds a DHCPv6 Solicit message. The Solicit may include a Rapid Commit option if a two-message exchange (i.e., instead of four) is required. The RS message may also include a Nonce option to provide an extended transaction identifier [RFC3971]. The node then sends the RS message either to the unicast address of a specific router on the link, or to the all-routers multicast address.

When a router receives an RS message with a DHCPv6 option, if it does not recognize the option and/or does not employ a DHCPv6 relay agent or server, it returns an RA message as normal with any stateless configuration information and without including a DHCPv6 option. By receiving the RA message with no DHCPv6 option, the node can determine that the router does not recognize the option and/or does not support a DHCPv6 relay/server function. In this way, no harm will have come from the node including the DHCPv6 option in the RS, and the function is fully backwards compatible.

When a router receives an RS message with a DHCPv6 option, if it recognizes the option and employs a DHCPv6 relay agent or server, it extracts the encapsulated DHCPv6 message and forwards it to the relay agent or server. When the DHCPv6 message reaches a DHCPv6 server, the server processes the DHCPv6 Solicit message and prepares either an Advertise (four message) or Reply (two message) DHCPv6 message containing any delegated addresses, prefixes and/or any other information the server is configured to send. The server then returns the Advertise/Reply message to the router.

When the router receives the DHCPv6 Advertise/Reply message, it creates a Router Advertisement (RA) message that includes any autoconfiguration information necessary for the link and also embeds the DHCPv6 message in a DHCPv6 option within the body of the RA. (The RA also echos the Nonce value if a Nonce was included in the RS message.) The router then returns the RA as a unicast message response to the node that sent the RS.

In a two message exchange, the stateless/stateful exchange is completed when the node receives the RA. In a four message exchange, the requesting node can Decline any stateful information it does not wish to accept and/or send unicast Request options in subsequent RSes to get RA messages with Reply options back from the router or routers of its choosing.

At any time after the initial RS/RA exchange, the node may need to issue DHCPv6 Renew, Release or Rebind messages to manage address/prefix lifetimes. In that case, the node prepares a DHCPv6 message option and inserts it in an RS message which it then sends via unicast to the router. The router in turn processes the message the same as for DHCPv6 Solicit/Reply.

At any time after the initial RS/RA exchange, the DHCPv6 server may need to issue a DHCPv6 Reconfigure message. In that case, when the router receives the DHCPv6 Reconfigure message it prepares a unicast RA message with a DHCPv6 option that encodes the Reconfigure and sends the RA as an unsolicited unicast message to the node. The node then follows the DHCPv6 client procedures for processing and responding to Reconfigure messages.

At any time after the initial RS/RA exchange, the router can initiate an unsolicited RA/Reply, e.g., to cause the node to update its configuration information quickly. In this method, the router sends a synthesized DHCPv6 Renew or Information-request message that induces the server to return a DHCPv6 Reply. The message includes the same DHCPv6 transaction-id and IPv6 ND Nonce values that the router had echoed in its initial Reply. The server then wraps the Reply message in the body of an RA message, and sends the unsolicited RA/Reply. When the node receives the unsolicited RA/Reply message, it matches the transaction-id and Nonce values with the initial RA/Reply it had received from the router. If the identification information matches, the node processes the message and initiates a new RS/RA exchange if necessary; otherwise it drops the message.

2.3. Stateful Provisioning Requirements

Using the DHCPv6 Option, the message itself includes sub-options to request stateful information. The DHCPv6 Device Unique Identifier (DUID) provides the identity of the requesting node, and the DHCPv6 transaction-id and IPv6 ND Nonce provide a unique identifier for matching RS and RA messages. Finally, the message can be protected using SEcure Neighbor Discovery (SEND) [RFC3971].

2.4. Implementation Considerations

The IPv6ND and DHCPv6 functions are typically implemented in separate router modules. In that case, the IPv6ND function extracts the DHCPv6 message from the option included in the RS message and wraps it in IP/UDP headers with the same addresses and port numbers the soliciting node would have used had it send an ordinary IP/UDP/DHCPv6 message. The IPv6ND function then acts as a Lightweight DHCPv6 Relay Agent (LDRA) [RFC6221] to forward the message to the DHCPv6 relay or server function on-board the router.

The forwarded DHCPv6 message then traverses any additional relays on the reverse path until it reaches the DHCPv6 server. When the DHCPv6 server processes the message, it delegates any necessary resources and returns a Reply via the same relay agent path as had occurred on the reverse path so that the Reply will eventually arrive back at the IPv6ND function. The IPv6ND function then prepares an RA message with any autoconfiguration information associated with the link, embeds the DHCPv6 message body in an IPv6ND DHCPv6 option, and returns the message via unicast to the node that sent the RS.

In an ideal implementation, the IPv6ND and DHCPv6 functions could be co-located in the same module on the router. In that way the two functions would be coupled as though they were in fact a single unified function without the need for any LDRA processing.

3. PIO Options in RS Messages

The second method entails the inclusion of Prefix Information Options (PIOs) in IPv6ND RS messages, as discussed in the following sections.

3.1. The PIO Option in RS Messages

This document proposes the inclusion of PIOs in RS messages to solicit and maintain prefixes that are delegated in subsequent RA messages. Prefix management is performed as discussed in [I-D.naveen-slaac-prefix-management] (an alternate prefix management proposal based on unsolicited advertisements with special flag settings is found in [I-D.pioxfolks-6man-pio-exclusive-bit]).

3.2. PIO Option Usage

When a node that wishes to request a prefix delegation first comes onto the link, it creates an RS message containing a PIO. It sets the Prefix Length to either the length of the prefix it wishes to receive or '0' (unspecified) if it will defer to the router's preference. The node then sets the Valid and Preferred Lifetimes to either its preferred values or '0' (unspecified) if it will defer to

the router's preference. The node then sets the Prefix to either the prefix it wishes to receive, or '0' (unspecified) if it will defer to the router's preference. The node then sends the RS message either to the unicast address of a specific router on the link, or to the all-routers multicast address.

When a router receives an RS message with a PIO, if it is not configured to accept PIOs in RS messages it returns an RA message as normal and without including a PIO. By receiving the RA message with no PIO, the node can determine that the router does not recognize the option and/or does not support an IPv6ND-based prefix delegation service. In this way, no harm will have come from the node including the PIO in the RS, and the function is fully backwards compatible.

When a router receives an RS message with a PIO, if it is configured to accept the option and can provide prefix delegation services it examines the fields in the message and selects a prefix to delegate to the node. If the PIO included a specific Prefix, the router delegates the node's preferred prefix if possible. Otherwise, the router selects a prefix to delegate to the node with length based on the node's Prefix Length. The router sets lifetimes matching the lifetimes requested by the node if possible, or shorter lifetimes if the node's requested lifetimes are too long. The router finally prepares a PIO containing this information and inserts it into an RA message to send back to the source of the RS.

3.3. Stateful Provisioning Requirements

Using the PIO in RS messages, the option itself requests stateful information. The RS message link-layer address can be used as the identity of the requesting node. The RS message includes a Nonce option [RFC3971] to provide a transaction identifier for matching RS and RA messages. Finally, the message can be protected using SEND the same as for the DHCPv6 option.

3.4. Implementation Considerations

Each router can implement a stateful database management service of their own choosing, but a functional alternative would be to use the standard DHCPv6 service as the back-end management service. In this way, all communications between the router's link to the requesting node are via RS/RA messaging. But, when the router receives an RS message with a PIO it can create a synthesized DHCPv6 Solicit message to send to the DHCPv6 server. This can be done in the same way as for the approach discussed in Section 2.4. In this way, the node on the link over which the PIO is advertised only ever sees RS/RA messages on the front end, and the router gets to use the DHCPv6 service for stateful configuration management on the back end.

4. Embedded Prefix Assertion

The third method entails a simple RS/RA exchange with no additional options where the node asserts (or "registers") a prefix by embedding the prefix in the source address of the RS message. The following sections provide further details.

4.1. Embedded Prefix Assertion

In this method, the node is pre-provisioned with the prefix it will use on its downstream networks (e.g., through network management, manual configuration, etc.). To invoke this method, the node includes its pre-provisioned prefix in the link-local source address of its RS message according to the OMNI address format [I-D.templin-6man-omni-interface]. For example, if the node is pre-provisioned with the prefix 2001:db8:1000:2000::/64, it creates its IPv6 link-local source address as fe80:2001:db8:1000:2000::.

4.2. Embedded Prefix Usage

When a node that wishes to assert a prefix first comes onto the link, it statelessly configures an OMNI address based on its pre-provisioned prefix. The node then includes the OMNI address as the source address of a standard RS message. If a router that receives the RS message has a way of verifying that the node is authorized to receive the asserted prefix, the router injects the prefix into the routing system and returns a standard RA message. When the node receives the RA message, it then has assurance that the proper routing state has been established.

The node examines the default router lifetime in the RA message as guidance for when subsequent RS/RA exchanges are necessary, i.e., the same as for normal IPv6ND. The node sends additional RS messages before the default router lifetime expires in order to keep the prefix assertion alive in the network. The RS messages may be sent either to the all-routers multicast address, to the link-local subnet router anycast address (i.e., fe80::) or to the unicast address(es) of the router(s) discovered through means outside the scope of this document.

4.3. Stateful Provisioning Requirements

Using embedded prefix assertion, the network must have some way of determining the node's authority to assert its claimed prefix. This could be, e.g., through examination of the link-layer source address of the RS message. The network must also have some way of knowing the node's claimed prefix length, as the length cannot be conveyed in the RS message. If necessary, the exchange can also include some

form of transaction identifier, e.g., by including a Nonce option in the RS. Finally, the exchange can be protected using SEND the same as for the previous two methods.

4.4. Implementation Considerations

This method can be conducted using standard RS/RA messages. It entails an administrative assignment of the node's OMNI address to the upstream interface over which it will send the RS. When the router receives the standard RS message, it statelessly derives the node's prefix from the OMNI address and injects the prefix into the routing system. The router then returns a standard RA message.

When the router returns the RA message, if it is configured to do so it can include a PIO option as discussed in Section 3.1. The PIO option includes prefix lifetimes and the prefix length. This "hybrid" combination of methods two and three could be useful in some deployment scenarios.

The same as for the PIO-based service discussed in Section 3.4, DHCPv6 can be used as the back-end service for stateful configuration management.

5. Out-of-Band Network Login Messaging

The fourth method entails an out-of-band messaging exchange through a "network login" procedure. During the network login, the requesting node could have an out-of-band messaging exchange with the network to prepare for the router eventually sending an RA message as discussed in the following sections

5.1. Out-of-Band Network Login

In the out-of-band network login, the node signs into the network using, e.g., a login/password, a security certificate, etc. The node authenticates itself to the network, and can optionally have an iterative exchange to request certain aspects of the node's desired stateful configuration information. The first-hop router is then signaled to prepare an RA message to return to the node, i.e., either through some out-of-band signaling or through the node sending an RS message.

5.2. Out-of-Band Network Login Usage

When a node first comes onto the link, it engages in a network login session using some form of out-of-band messaging such as Layer-2 (L2) messaging. The session entails a security exchange where the node authenticates itself to the network and proves its authorization to

receive the stateful configuration information. The network then signals the router to send an RA message to the node, either unsolicited or in response to the node's RS message.

5.3. Stateful Provisioning Requirements

Using out-of-band messaging, the node engages in an iterative exchange where a request for stateful configuration information is conveyed. The exchange includes an identity for the requesting node and provides a unique per-message identifier so that the node can correlate its message requests with the responses it gets back from the network. Finally, the message exchange itself contains security parameters for authenticating the requesting node.

5.4. Implementation Considerations

The network login system and routers must be tightly coupled so that the network login can securely convey the requesting node's identity to the router.

As for the PIO-based service discussed in Section 3.4, DHCPv6 can be used as the back-end service for managing the stateful configuration database.

6. Implementation Status

The approach discussed in Section 2 has been implemented as extensions to the OpenVPN open source software distribution. The implementation is available at: <http://linkupnetworks.net/aero/AERO-OpenVPN-2.0.tgz>.

7. IANA Considerations

The IANA is instructed to assign an IPv6ND option Type value TBD for the DHCPv6 option.

The IANA is instructed to create a registry for the DHCPv6 option "Reserved" field (with no initial assignments) so that future uses of the field can be coordinated.

8. Security Considerations

Security considerations for IPv6 Neighbor Discovery [RFC4861] and DHCPv6 [RFC8415] apply to this document.

SEcure Neighbor Discovery (SEND) [RFC3971] can provide authentication for IPv6 ND messages with no need for additional securing mechanisms.

9. Acknowledgements

This work was motivated by discussions on the 6man and v6ops list. Those individuals who provided encouragement and critical review are acknowledged.

The following individuals provided useful comments that improved the document: Mikael Abrahamsson, Fred Baker, Ron Bonica, Yucel Guven, Naveen Kottapalli, Ole Troan, Bernie Volz.

The following individuals developed IPv6ND and DHCPv6 extensions for OpenVPN: Kyle Bae, Wayne Benson, Eric Yeh.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the FAA as per the SE2025 contract number DTFWA-15-D-00030.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program and the Boeing Research & Technology (BR&T) enterprise autonomy program.

10. References

10.1. Normative References

- [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>>.
- [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>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.

10.2. Informative References

- [I-D.naveen-slaac-prefix-management]
Kottapalli, N., "IPv6 Stateless Prefix Management", draft-naveen-slaac-prefix-management-00 (work in progress), November 2018.
- [I-D.pioxfolks-6man-pio-exclusive-bit]
Kline, E. and M. Abrahamsson, "IPv6 Router Advertisement Prefix Information Option eXclusive Flag", draft-pioxfolks-6man-pio-exclusive-bit-02 (work in progress), March 2017.
- [I-D.templin-6man-omni-interface]
Templin, F. and T. Whyman, "Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces", draft-templin-6man-omni-interface-67 (work in progress), December 2020.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC6221] Miles, D., Ed., Ooghe, S., Dec, W., Krishnan, S., and A. Kavanagh, "Lightweight DHCPv6 Relay Agent", RFC 6221, DOI 10.17487/RFC6221, May 2011, <<https://www.rfc-editor.org/info/rfc6221>>.

Appendix A. Change Log

<< RFC Editor - remove prior to publication >>

Changes from -09 to -10:

- o Changed "AERO" reference to "OMNI"
- o Version number and reference update.

Changes from -08 to -09:

- o Changed reference and draft name for Prefix Assertion / Registration

Changes from -07 to -08:

- o Changed DHCPv6 reference to RFC8415 - deprecates RFC3315 and RFC3633
- o added prefix length to example in Section 4.1.

Changes from -06 to -07:

- o Added "unsolicited DHCPv6 Reply" considerations
- o Added refeence to new IPv6ND-based PD proposal.
- o No longer associate the term "autoconfiguration" with the term "stateful".
- o Added URL for implementation.

Author's Address

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

IPv6 Maintenance
Internet-Draft
Updates: 4862 (if approved)
Intended status: Standards Track
Expires: January 1, 2019

L. Colitti
E. Kline
Google
June 30, 2018

Zero valid lifetimes on point-to-point links
draft-zero-rafols-6man-ra-zero-lifetime-00

Abstract

This document allows implementations to accept low or zero valid lifetimes in Router Advertisement Prefix Information Options in cases where it is known that there can only be one router on the link.

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 January 1, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Requirements Language	2
2. Cases when it is useful to reduce Valid Lifetime to zero . .	2
3. Changes to RFC 4862	2
4. IANA Considerations	3
5. Security Considerations	3
6. References	3
6.1. Normative References	3
6.2. Informative References	3
Authors' Addresses	3

1. Introduction

Currently, Prefix Information Options in Router Advertisements cannot reduce the Valid Lifetime of an IPv6 address below 2 hours. This is due to an explicit restriction in Section 5.5.3 of [RFC4862]. The reason is to avoid a denial-of-service attack whereby a malicious attacker can cause a node's addresses to expire prematurely by sending a Router Advertisement with a low Valid Lifetime.

1.1. Requirements Language

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

2. Cases when it is useful to reduce Valid Lifetime to zero

In some cases, it is useful for the network to inform the host that a given prefix is no longer valid or will shortly no longer be valid. One example is if the host has moved beyond the mobility scope of the prefix and the network will no longer deliver packets for that prefix to the host. The host can thus terminate any upper-layer connections using that prefix and notify applications that continued communication will require using a new source address.

In order to ensure uninterrupted communications and no disruption to applications, this should be done only if the host already has other IPv6 addresses of equivalent scope and sufficient Valid Lifetime.

3. Changes to RFC 4862

The following clause is added between points 1 and 2 of clause e, Section 5.5.3 of [RFC4862]:

2. If the link-layer guarantees that there is only one node on the link from which the host can receive Router Advertisements (e.g., if the link is a point-to-point link, such as a PPP link or a 3GPP link as defined in [RFC6459]), and the link has another prefix of the same scope with sufficient Valid Lifetime, set the valid lifetime of the corresponding address to the advertised Valid Lifetime.

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

The denial-of-service attack that motivated this restriction cannot be mounted on a link where no other devices can send Router Advertisements to the host.

6. References

6.1. 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>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.

6.2. Informative References

- [RFC6459] Korhonen, J., Ed., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, DOI 10.17487/RFC6459, January 2012, <<https://www.rfc-editor.org/info/rfc6459>>.

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