Abstract

This document describes stateless best effort Multicast along the shortest paths to the egress nodes of a P2MP Path/Tree. The multicast data packet is encapsulated in an IPv6 Multicast Routing Header (MRH). The MRH contains the egress nodes represented by the indexes of the nodes and flexible bit strings for the nodes. The packet is delivered to each of the egress nodes along the shortest path. There is no state stored in the core of the network.

Requirements Language

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

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/.
The potential egress nodes and transit nodes in a network are numbered or indexed from 1 to the number of the nodes. Figure 1 shows an example network having nodes E1 to PE10 and P1 to P5, where PE1 to PE10 are edge nodes (i.e., potential egress nodes) and P1 to
P5 are transit nodes. In this example, these nodes have node indexes 1 to 10 and 11 to 15 respectively. The number labeling a link is the cost of the link. For example, 5 on the link between P5 and PE4 is the cost of the link. The cost of a link without a numeric label is 1.

![Network Diagram]

Figure 1: Network with 10 edges and P2MP tree from PE1 to PE2 - PE6

The P2MP path/tree from ingress PE1 towards egresses PE2 to PE6 (i.e., PE2, PE3, PE4, PE5 and PE6) in Figure 1 is represented by the set of the indexes of the egress nodes of the P2MP path. The indexes of PE2 to PE6 are 2 to 6 (i.e., 2, 3, 4, 5 and 6) respectively. These indexes are encoded by the indexes directly or the bit string of one byte.

A controller such as PCE as a controller can have the information about the node indexes, and send the P2MP path to the ingress of the path.

After receiving a data packet from traffic source CE1, ingress PE1 encapsulates the packet in a MRH with the P2MP path represented by the indexes. The packet is transmitted along the shortest path to each of the egresses.
This document describes the encoding of a P2MP Path/Tree using the indexes of the egress nodes of the tree and specifies the procedure/behavior of the nodes along the shortest paths to the egresses.

1.1. Acronyms

The following acronyms are used in this document:

CE: Customer edge/equipment.
MRH: Multicast Routing Header.
P2MP: Point 2 Multi-Point.
PE: Provider Edge.

2. Encoding of P2MP Path/Tree

A simple encoding could use a fixed length field such as 2 bytes to store a node index. The P2MP path/tree from ingress PE1 to egresses PE2 – PE6 (i.e., PE2, PE3, PE4, PE5 and PE6) in Figure 1 is represented in Figure 2. There are five fields, each of which occupies 2 bytes and stores the index of an egress node. These five fields store 2, 3, 4, 5 and 6, which are the indexes of egress nodes PE2 to PE6 respectively.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               2               | PE2’s Index
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               3               | PE3’s Index
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               4               | PE4’s Index
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               5               | PE5’s Index
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               6               | PE6’s Index
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Encoding tree from PE1 to PE2 - PE6 with indexes directly

2.1. More Efficient Encoding of P2MP Path/Tree

However, in many cases, we can improve on the above encoding by having a B flag to indicate whether an entry uses a bit string to represent multiple node indexes. When B = 0, there is one fixed length (15 bits) field following B, having a node index. When B = 1, there are three fields following B: a start index (StartIndex) field,
a size of bit string (S-BitString) field and a bit string (BitString) field. The StartIndex field contains a starting node number (index) as an unsigned integer. The S-BitString field of 1 byte indicates the size of the BitString field in bytes as an unsigned integer. The BitString field contains a bit string, where each bit with value of 1 in the string indicates a node index equal to the start index plus that bit position number.

The P2MP path/tree from ingress PE1 to egresses PE2 - PE6 (i.e., PE2, PE3, PE4, PE5 and PE6) in Figure 1 is represented in Figure 3 using a bit string.

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 0 1 2 3 4 5 6 7 1 2 3 4 5 6 7 8
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|1|              1              |       1       |1|1|1|1|0|0|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|B|          StartIndex         |  S-BitString  |   BitString   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

Figure 3: Encoding tree from PE1 to PE2 - PE6 using bit string

The indexes of egress nodes PE2 to PE6 are represented by four fields: B flag with value of 1, StartIndex of 15 bits with value of 1, the S-BitString byte with value of 1, and the BitString of 1 byte (i.e., 8 bits) with value 0b1111000.

S-BitString = 1 indicates BitString occupies 1 byte. BitString = 0b1111000 combined with StartIndex = 1 indicates five node indexes 2, 3, 4, 5 and 6.

The BitString’s first bit = 1 indicates the first node index after the start index 1, which is 2 (i.e., 2 = 1 + 1); the BitString’s second bit = 1 indicates the second node index after the start index 1, which is 3 (i.e., 3 = 1 + 2); and so on.

Suppose that the index of egress node PE2 is 2 and the indexes of egress nodes PE3 to PE6 are 30003 to 30006 respectively. The P2MP path/tree from ingress PE1 to egresses PE2 - PE6 can be represented as shown in Figure 4 using the node index and bit string.
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| NodeIndex = 2 | PE2’s Index 1 2 3 4 5 6 7 8
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1| 30002 | 1 |1|1|1|0|0|0|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|B| StartIndex | S-BitString | BitString |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Figure 4: Encoding tree from PE1 to PE2 – PE6 using index and bit string

The node index of PE2 is represented by B = 0 and NodeIndex of 15 bits with value of 2 (i.e., NodeIndex = 2).

The indexes of egress nodes PE3 to PE6 are represented by four fields: B flag of 1 bit with value of 1, StartIndex of 15 bits with value of 30002, S-BitString of 8 bits with value of 1, and BitString of 1 byte (i.e., 8 bits) with value 0b11110000. S-BitString = 1 indicates BitString occupies 1 byte. BitString = 0b11110000 combined with StartIndex = 30002 indicates four node indexes 30003 to 30006.

The BitString’s first bit = 1 indicates the first node index after StartIndex = 30002, which is 30003 (i.e., 30003 = 30002 + 1); The BitString’s second bit = 1 indicates the second node index after StartIndex = 30002, which is 30004 (i.e., 30004 = 30002 + 2); and so on.

3. Node Index Table

Every node in a network has a node index IPv6 table. The table has a row for the index of each egress node with the IPv6 address and the index of the next hop on the shortest path to that node. This table indicates the shortest IGP path to each egress, i.e., the next hop of the shortest path to each egress. This is similar to a unicast forwarding table but organized by exact match node index rather than longest match IP address or the like. Figure 5 shows an example Node Index IPv6 table of PE1 in Figure 1.
<table>
<thead>
<tr>
<th>Node index</th>
<th>IPv6 Address of next hop</th>
<th>Index of next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>2</td>
<td>IPv6 address of P1</td>
<td>11 (index of P1)</td>
</tr>
<tr>
<td>3</td>
<td>IPv6 address of P1</td>
<td>11 (index of P1)</td>
</tr>
<tr>
<td>4</td>
<td>IPv6 address of P1</td>
<td>11 (index of P1)</td>
</tr>
<tr>
<td>5</td>
<td>IPv6 address of P1</td>
<td>11 (index of P1)</td>
</tr>
<tr>
<td>6</td>
<td>IPv6 address of P1</td>
<td>11 (index of P1)</td>
</tr>
<tr>
<td>7</td>
<td>IPv6 address of P1</td>
<td>11 (index of P1)</td>
</tr>
<tr>
<td>8</td>
<td>IPv6 address of P1</td>
<td>11 (index of P1)</td>
</tr>
<tr>
<td>9</td>
<td>IPv6 address of P1</td>
<td>11 (index of P1)</td>
</tr>
<tr>
<td>10</td>
<td>IPv6 address of PE10</td>
<td>10 (index of PE10)</td>
</tr>
</tbody>
</table>

Figure 5: Node Index IPv6 Table of PE1

The table has 10 rows of node index and IPv6 address. The 10 rows have node indexes of egress nodes PE1 to PE10, the IPv6 addresses and the indexes of the next hops of the shortest paths to PE1 to PE10 respectively. The next hop to PE1 itself is NULL. The next hop to each of PE2 to PE9 is P1. The next hop to PE10 is PE10. Note: The information such as port number or interface used to forward a packet to the next hop is not shown in the figure, which is the same as the corresponding information in the forwarding table (FIB) of PE1.

For example, the second row has node index 2 of PE2 and the IPv6 address of the next hop node to PE2, which is IPv6 address of P1 since the next hop to PE2 is P1. The tenth row has node index 10 of PE10 and the IPv6 address of the next hop to PE10, which is IPv6 address of PE10 since the next hop to PE10 is PE10.

Figure 6 shows an example Node Index IPv6 table of P1 in Figure 1. The table has 10 rows of node index and IPv6 address. The 10 rows have node indexes of PE1 to PE10 and the IPv6 addresses of the next hops of the shortest paths to PE1 to PE10 respectively.
<table>
<thead>
<tr>
<th>Node index</th>
<th>IPv6 Address of next hop</th>
<th>Index of next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IPv6 address of PE1</td>
<td>1 (index of PE1)</td>
</tr>
<tr>
<td>2</td>
<td>IPv6 address of P2</td>
<td>12 (index of P2)</td>
</tr>
<tr>
<td>3</td>
<td>IPv6 address of P2</td>
<td>12 (index of P2)</td>
</tr>
<tr>
<td>4</td>
<td>IPv6 address of P5</td>
<td>15 (index of P5)</td>
</tr>
<tr>
<td>5</td>
<td>IPv6 address of P5</td>
<td>15 (index of P5)</td>
</tr>
<tr>
<td>6</td>
<td>IPv6 address of P5</td>
<td>15 (index of P5)</td>
</tr>
<tr>
<td>7</td>
<td>IPv6 address of P5</td>
<td>15 (index of P5)</td>
</tr>
<tr>
<td>8</td>
<td>IPv6 address of PE8</td>
<td>8 (index of PE8)</td>
</tr>
<tr>
<td>9</td>
<td>IPv6 address of PE9</td>
<td>9 (index of PE9)</td>
</tr>
<tr>
<td>10</td>
<td>IPv6 address of PE1</td>
<td>1 (index of PE1)</td>
</tr>
</tbody>
</table>

Figure 6: Node Index IPv6 Table of P1

For example, the first row has node index 1 of PE1 and the IPv6 address of the next hop node to PE1, which is IPv6 address of PE1 since the next hop to PE1 is PE1.

The second row has node index 2 of PE2 and the IPv6 address of the next hop node to PE2, which is IPv6 address of P2 since the next hop to PE2 is P2.

The fourth row has node index 4 of PE4 and the IPv6 address of the next hop node to PE4, which is IPv6 address of P5 since the next hop to PE4 is P5.

The tenth row has node index 10 of PE10 and the IPv6 address of the next hop to PE10, which is IPv6 address of PE1 since the next hop to PE10 is PE1.

4. IPv6 Multicast Routing Header (MRH)

Figure 7 shows a Multicast Routing Header (MRH) in an IPv6 packet. The IPv6 packet has an IPv6 header with a destination address (DA) and source address (SA) of IPv6, a routing header with Routing type
(TBD) indicating MRH and an IP multicast datagram. The routing header is indicated by the Next Header in the IPv6 header.

|<--IPv6 header-->|<-Routing header->|
+-----------------+-----------------+------------------------+
| Next Header =  | Next Header      | (an extension header)  |
| Routing header |      |                  |                        |
| SA=IPv6 Address| Routing Type =   | IP multicast datagram  |
| DA=IPv6 Address|   TBD (MRH)      |                        |
|                 | SL, NE, Sub-tree|                        |
+-----------------+-----------------+------------------------+

Figure 7: Multicast Routing Header (MRH) in IPv6 packet

The format of the MRH is shown in Figure 8.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Hdr Ext Len  |RoutingType=TBD|Version| Flags |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  SL (Subtree Left)  |NE (# Egresses)|        Reserved         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Sub-tree encoding of node indexes               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Format of Multicast Routing Header (MRH)

The MRH has the following fields:

Next Header: The type of the header after the MRH. Either another extension header or the type of IP multicast datagram in the packet.

Hdr Ext Len: Its value indicates the length of the MRH in a unit of 64 bits (i.e., 8 bytes) excluding the first 8 bytes.

Routing Type: Its value TBD indicates that the routing header is a Multicast Routing Header (MRH).

Version: The Version of the MRH. This document specifies Version zero.

Flags: No flag is defined yet.

Sub-tree Left (SL): Its value points to the sub-tree.
Number of Egresses (NE): Its value indicates the number of egress
nodes of the sub-tree.

Sub-tree: Its value encodes the egress nodes of the sub-tree. A
node index MUST NOT occur more than once. The node indexes in
sub-tree are ordered so that, the egress nodes in every sub-tree
of the P2MP tree are contiguous.

The P2MP path/tree from PE1 via P1 to PE2, PE3, PE4, PE5 and PE6 as
shown in Figure 1 is encoded by the indexes of the egress nodes of
the tree as illustrated in Figure 2. For an IP multicast datagram/
packet to be transmitted by the P2MP path/tree, PE1 constructs an
IPv6 packet for each sub-tree and sends the packet containing a MRH
and the IP multicast datagram/packet to the next hop along the sub-
tree.

The number of sub-trees from PE1 is the number of different next hop
nodes from PE1 to the egress nodes (i.e., PE2 to PE6). PE1 gets the
next hops to the egress nodes using its Node Index IPv6 Table as
shown in Figure 5 with the node indexes of the egress nodes, which
are 2, 3, 4, 5 and 6. The next hops are the same, which are P1.
Thus, there is one sub-tree from PE1 via P1 towards PE2 to PE6.

PE1 sets DA of the IPv6 packet to P1’s IPv6 address (P1’s IPv6 for
short) and SA of the packet to PE1’s IPv6 address (PE1’s IPv6 for
short). PE1 builds the MRH based on the encoding of the tree in
Figure 2 through including the sub-tree from P1 and setting SL to 10
as a pointer pointing to the sub-tree and setting NE to 5, which is
the number of egresses of the sub-tree. Figure 9 shows the packet to
be sent to P1, which is received by P1.
After receiving the IPv6 packet from PE1, P1 determines whether the packet’s next header is a MRH through checking if the next header is a routing header, and if so, whether the routing type in the routing header is TBD for MRH. When the next header is the MRH, P1 duplicates the packet for each sub-tree from P1 and sends the packet copy with an updated MRH to the next hop along the sub-tree.

P1 gets the next hops to the egress nodes using its Node Index IPv6 Table as shown in Figure 6 with the node indexes of the egress nodes, which are 2, 3, 4, 5 and 6. PE2 and PE3 have the same next hop P2 according to the table. PE4 to PE6 have the same next hop P5.

There are 2 sub-trees from P1. One sub-tree is from P1 via next hop P2 to PE2 and PE3. The other is from P1 via next hop P5 to PE4, PE5 and PE6. P1 duplicates the packet for each of these two sub-trees and sends the packet copy to the next hop along the sub-tree.

P1 sets the DA of one packet copy to P2’s IPv6 address. P1 updates the MRH based on the encoding of the tree in Figure 9 through setting NE to the number of egress nodes of the sub-tree from P2 to PE2 and PE3 (which is 2). Figure 10 shows the IPv6 packet to be sent to P2, which is received by P2.
Figure 10: IPv6 packet with MRH received by P2

P1 sets the DA of the other packet copy to P5’s IPv6 address. P1 updates the MRH based on the encoding of the tree in Figure 9 through setting SL to 6 as a pointer pointing to the start of the sub-tree from P5 to PE4, PE5 and PE6, setting NE to 3, which is the number of egress nodes of the sub-tree from P5 to PE4, PE5 and PE6. Figure 11 shows the IPv6 packet to be sent to P5, which is received by P5.

Figure 11: IPv6 packet with MRH received by P5

After receiving the IPv6 packet from P1, P5 determines whether the packet’s next header is an MRH. When the next header is an MRH, P5 duplicates the packet for each sub-tree from P5 and sends the packet copy with an updated MRH to the next hop along the sub-tree.
P5 gets the next hops to the egress nodes using its Node Index IPv6 Table with the node indexes of the egress nodes, which are 4, 5 and 6. PE4, PE5 and PE6 have the same next hop P4 according to the table.

P5 sets the DA of the packet copy to P4’s IPv6 address. P5 updates the MRH based on the encoding of the tree in Figure 11 through setting SL to 6, which points to the start of the sub-tree from P4 to PE4, PE5 and PE6, NE to the number of egress nodes of the sub-tree from P4 to PE4 and PE5 (which is 3). Figure 12 shows the packet to be sent to P4, which is received by P4.

```
| IPv6 Header | <------- MRH -------> |
+-------------+-----------------------+-------------+
| DA = P4’s IPv6 | RoutingType=TBD,SL=6,NE=3 | IP multicast |
| SA = PE1’s IPv6 | sub-tree from P4 to PE4-PE6 datagram |
```

Figure 12: IPv6 packet with MRH received by P4

After receiving the IPv6 packet from P5, P4 determines whether the packet’s next header is an MRH. When the next header is the MRH, P4 duplicates the packet for each sub-tree from P4 and sends the packet copy with an updated MRH to the next hop along the sub-tree.

P4 gets the next hops to the egress nodes using its Node Index IPv6 Table with the node indexes of the egress nodes, which are 4, 5 and 6. PE4, PE5 and PE6 are the next hops PE4, PE5 and PE6 themselves according to the table.

P4 sends the copy with MRH containing SL = 0 to each of PE4, PE5 and PE6. The packet received by PE4 is shown in Figure 13.

```
| IPv6 Header | <------- MRH -------> |
+-------------+-----------------------+-------------+
| DA = PE4’s IPv6 | RoutingType=TBD,SL=0,NE | IP multicast |
| SA = PE1’s IPv6 | datagram               |
```

Figure 13: IPv6 packet with MRH received by PE4
When a leaf/egress such as PE4 receives an IPv6 packet with MRH having SL = 0, the leaf/egress sends the IP multicast packet to the multicast layer of the leaf/egress.

5. Procedures at Nodes

This section describes the procedures at the ingress, transit and egress/leaf nodes of a P2MP path/tree delivering a packet received from the path to its destinations.

5.1. Procedure at Ingress Node

For a packet to be transported by a P2MP path/tree, the ingress of the P2MP path/tree duplicates the packet for each sub-tree/branch of the P2MP path/tree branching from the ingress, encapsulates each packet copy in a MRH containing its sub-tree and sends the encapsulated packet copy to the next hop node along that sub-tree.

For example, there is one sub-tree branching from the ingress of the P2MP path/tree in Figure 1. The sub-tree is from ingress PE1 via next hop node P1 towards PE2 to PE6. PE1 sends P1 the packet as shown in Figure 9.

5.2. Procedure at Transit Nodes

When a transit node receives a packet encapsulated in an MRH, the node executes the procedure to duplicate the packet for each of the sub-trees/branches from the transit node on the path/tree and sends the packet copy to the next hop along each sub-tree.

The number of sub-trees from the transit node is the number of different next hop nodes from the transit node to the egress nodes. The transit node gets the next hops to the egress nodes using its Node Index IPv6 Table with the node indexes of the egress nodes.

The transit node sets the DA of a packet copy to a next hop node’s IPv6 address. The transit node updates the MRH based on the encoding of the sub-tree in the packet received.

For example, after receiving the IPv6 packet as shown in Figure 9, the transit node P1 duplicates the packet for each branch/sub-tree from P1 and sends the packet copy with an updated MRH to the next hop along the branch/sub-tree.

P1 gets the next hops to the egress nodes using its Node Index IPv6 Table as shown in Figure 6 with the node indexes of the egress nodes, which are 2, 3, 4, 5 and 6 of PE2, PE3, PE4, PE5 and PE6.
respectively. PE2 and PE3 have the same next hop P2 according to the table. PE4, PE5 and PE6 have the same next hop P5.

There are 2 sub-trees from P1. One sub-tree is from P1 via next hop P2 to PE2 and PE3. The other is from P1 via next hop P5 to PE4, PE5 and PE6. P1 duplicates the packet for each of these two sub-trees and sends a packet copy to the next hop along each sub-tree.

P1 sets the DA of one packet copy to P2’s IPv6 address. P1 updates the MRH based on the encoding of the tree in Figure 9 through setting SL to 10 as a pointer pointing to the sub-tree from P2 to PE2 and PE3, and setting NE to 2, which is the number of egress nodes of the sub-tree. Figure 10 shows the packet to be sent to P2, which is received by P2. The other packet copy is updated in a similar fashion but for the other subtree.

5.3. Procedure at Egress Node

When an egress node of a P2MP path receives a packet transported by the path, the DA of the packet is the IPv6 address of the egress node and the MRH in the packet has SL = 0. In this case, the egress node decapsulates the packet and sends the IP multicast datagram to the IP multicast forwarding module.

For example, after receiving the IPv6 packet from P4 as shown in Figure 13, PE4 determines that the packet has a MRH with SL = 0. PE4 decapsulates the packet and sends the IP multicast datagram to the IP multicast forwarding module.

6. Security Considerations

For general IPv6 and IPv6 extension header security considerations, see [RFC8200]. More TBD

7. IANA Considerations

IANA is requested to assign a new Routing Type in the subregistry "Routing Types" under registry "Internet Protocol Version 6 (IPv6) Parameters" as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD (8 suggested)</td>
<td>Multicast Routing Header</td>
<td>This document</td>
</tr>
</tbody>
</table>
8. Acknowledgements

TBD

9. References

9.1. Normative References


9.2. Informative References


Authors’ Addresses

Huaimo Chen
Futurewei
Boston, MA
USA

Email: Huaimo.chen@futurewei.com
Xufeng Liu
IBM Corporation
USA

Email: xufeng.liu.ietf@gmail.com

Lei Liu
Fujitsu
USA

Email: liulei.kddi@gmail.com
Stateless Traffic Engineering Multicast using MRH
draft-chen-pim-mrh6-03

Abstract

This document describes a stateless traffic engineering (TE) multicast along an explicit P2MP Path/Tree using an IPv6 extension header called TE multicast routing header (MRH). The MRH with the path encoded in link numbers is added into a packet to be multicast at the ingress. The packet is delivered to the egresses along the path. There is no state stored in the core of the network.

Requirements Language

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

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 12, 2023.

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) 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. Overview of TE Multicast using MRH ...........................  3
3. Encoding of P2MP Path/Tree  ................................  5
4. Multicast Routing Header (MRH) ...............................  12
5. Procedures/Behaviors .......................................  19
   5.1. Procedure/Behavior on Ingress Node ....................  19
   5.2. Procedure/Behavior on Transit Node ....................  19
   5.3. Procedure/Behavior on Egress Node ....................  21
6. IANA Considerations .......................................  21
7. Security Considerations ....................................  21
8. Acknowledgements .........................................  22
9. References ...............................................  22
   9.1. Normative References ..................................  22
   9.2. Informative References ................................  22
Authors’ Addresses ............................................  22

1. Introduction

[I-D.ietf-pim-sr-p2mp-policy] proposes a solution for a SR P2MP path. A multicast P2MP tree is created for the path and the state of the
tree is instantiated in the forwarding plane by a controller at Root, intermediate Replication nodes and Leaves of the tree.

[I-D.chen-pim-srv6-p2mp-path] proposes a stateless solution for a SR P2MP path. The overhead for encoding the P2MP path using SIDs in SRH may be large.

This document describes a stateless traffic engineering (TE) multicast along an explicit P2MP Path/Tree using an IPv6 extension header called TE multicast routing header (MRH). The MRH with the path encoded in link numbers is added into a packet to be multicast at the ingress. The packet is delivered to the egresses along the path. There is no state stored in the core of the network.

2. Overview of TE Multicast using MRH

Figure 1 shows an example network having nodes P1, P2, P3, P4, PE1 to PE10. For each of the links connected to a node, a number called local link number (or link number for short) is assigned to it. For example, PE1 has three links: link from PE1 to CE1, link from PE1 to P1, and link from PE1 to PE10. These three links have link numbers 1, 2 and 3 respectively on PE1. P1 has five links: P1 to PE1, P1 to P2, P1 to P3, P1 to PE8 and P1 to PE9. These five links have link numbers 1, 2, 3, 4 and 5 respectively on P1. P3 has two links: P3 to P1 and P3 to P4. These two links have link numbers 1 and 2 respectively on P3.

![Diagram of network with P2MP Path from PE1 to PE2, PE3, ..., PE9]
The P2MP path from ingress PE1 via P1 towards egresses PE2 to PE9 in Figure 1 is represented by the link numbers along the path: PE1’s link number 2; P1’s link numbers 2, 3, 4 and 5; P2’s link numbers 1 and 2; P3’s link number 2; and P4’s link numbers 2, 3, 4 and 5.

A controller such as PCE as a controller has the link numbers of the links originating at every node. The controller can send the ingress the P2MP path represented by the link numbers of the links along the path.

After receiving a packet from traffic source CE1, ingress PE1 encapsulates the packet in a MRH with a P2MP path represented by the link numbers along the path. The packet in the MRH is transmitted along the path to the egresses of the path.

When a node such as P1 receives a packet with the MRH, the node gets/pops each of its link numbers, finds the address of the next hop from a neighbor address table using the link number (i.e., the link number such as 3 of the link from the node to the next hop such as P3), and sends the packet to the next hop (such as P3).

Figure 2 shows the neighbor IPv6 address table of P1. The table has five rows of link number, link type and IPv6 address for the five links of P1. The first row has link number 1, link type P2P for link from P1 to next hop PE1 and PE1’s IPv6 address. The 2nd row has link number 2, link type P2P for link from P1 to next hop P2 and P2’s IPv6 address. The 3rd row has link number 3, link type P2P for link from P1 to next hop P3 and P3’s IPv6 address. The 4-th row has link number 4, link type P2P for link from P1 to next hop PE8 and PE8’s IPv6 address. The 5-th row has link number 5, link type P2P for link from P1 to next hop PE9 and PE9’s IPv6 address.

<table>
<thead>
<tr>
<th>Link number</th>
<th>Link type</th>
<th>Address of next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P2P</td>
<td>IPv6 address of PE1</td>
</tr>
<tr>
<td>2</td>
<td>P2P</td>
<td>IPv6 address of P2</td>
</tr>
<tr>
<td>3</td>
<td>P2P</td>
<td>IPv6 address of P3</td>
</tr>
<tr>
<td>4</td>
<td>P2P</td>
<td>IPv6 address of PE8</td>
</tr>
<tr>
<td>5</td>
<td>P2P</td>
<td>IPv6 address of PE9</td>
</tr>
</tbody>
</table>

Figure 2: Neighbor IPv6 address table of P1
Using link number 1, P1 gets PE1’s IPv6 address from the table; using link number 2, P1 gets P2’s IPv6 address from the table; using link number 3, P1 gets P3’s IPv6 address from the table; and so on.

3. Encoding of P2MP Path/Tree

Figure 3 shows the encoding of the P2MP path/tree in Figure 1 from PE1 via P1 to PE2, PE3, ..., PE9. Each link on the tree is encoded or represented by the link number of the link. For example, the link from PE1 to P1 on the tree is encoded by link number 2 of the link on PE1. The link from P1 to P2 on the tree is encoded by link number 2 of the link on P1. The link from P1 to P3 on the tree is encoded by link number 3 of the link on P1.

For each link from its upstream node to its downstream (or say next hop) node on the tree, three fields are used for the link: 1). link number (Link-No for short) field for storing the link number on the upstream node, 2). number of branches (N-Branches for short) for storing the number branches/sub-trees of the tree from the next hop node, and 3). size of branches+ (S-Branches+ for short) for storing the size of branches of the tree from the next hop node and the fields following. The size is in bytes.
<table>
<thead>
<tr>
<th>size</th>
<th>Link-No</th>
<th>N-Branches</th>
<th>S-Branches+</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>2</td>
<td>4</td>
<td>22</td>
<td>PE1 to P1</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>2</td>
<td>14</td>
<td>P1 to P2 from PE1 to P1</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>P1 to P3 to PE2-PE9</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>P1 to PE8</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>P1 to PE9</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>P2 to PE2 sub-trees from P2</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>P2 to PE3</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>P3 to P4 sub-tree from P3</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>P4 to PE4</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>P4 to PE5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>P4 to PE6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>P4 to PE7</td>
</tr>
</tbody>
</table>

Figure 3: Encoding of tree from PE1 via P1 to PE2, PE3, ..., PE9

For example, for the link from PE1 to P1 on the tree, Link-No field has value of 2 since the link number is 2 for the link on PE1; N-Branches field has value of 4 since there are 4 branches from P1; S-Branches+ field has value of 22 (bytes) since 22 bytes are used for storing/encoding 4 branches from P1 when the three fields Link-No, N-Branches and S-Branches+ for a link occupy 2 bytes in total.

For the link from P1 to P2 on the tree, Link-No field has value of 2 since the link number is 2 for the link on P1; N-Branches field has value of 2 since there are two branches from P2; S-Branches+ field has value of 14 (bytes) since 14 bytes are used for storing/encoding the two branches (i.e., sub-trees) from P2 and the fields following.

For the link from P1 to P3 on the tree, Link-No field has value of 3 since the link number is 3 for the link on P1; N-Branches field has value of 1 since there is one branch (i.e., sub-tree) from P3; S-Branches+ field has value of 10 (bytes) since 10 bytes are used for storing/encoding the branch from P3 and no other fields following.
For the link from P2 to egress PE2 on the tree, Link-No field has value of 1 since the link number is 1 for the link on P2; N-Branches field has value of 0 since there is no branch (i.e., sub-tree) from PE2; S-Branches+ field has value of 0 (bytes).

Figure 4 shows an enhancement of encoding the P2MP path/tree in Figure 1 using L flag. The L flag added for a link indicates whether the next hop is a leaf node. When L is set to one indicating the next hop is a leaf (i.e., egress), the "N-Branches" and "S-Branches+" fields are removed. There are 8 links to leaves (i.e., egress nodes) on the tree, which are 2 links from P2 to PE2 and PE3; 4 links from P4 to PE4, PE5, PE6 and PE7; and 2 links from P1 to PE8 and PE9. These 8 links have their L flags set to one. The N-Branches and S-Branches+ fields for these 8 links are removed from the encoding of the tree. This reduces the space/memory for encoding the tree.

<table>
<thead>
<tr>
<th>size</th>
<th>L</th>
<th>Link-No</th>
<th>N-Branches</th>
<th>S-Branches+</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>14</td>
<td>PE1 to P1</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>P1 to P2</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>P1 to P3</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td>P1 to PE8</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td>P1 to PE9</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>P2 to PE2</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td>P2 to PE3</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>P3 to P4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td>P4 to PE4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td>P4 to PE5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td>P4 to PE6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td>P4 to PE7</td>
</tr>
</tbody>
</table>

Figure 4: Encoding of tree from PE1 via P1 to PE2, PE3, ..., PE9 with L flag
For example, suppose that two fields L and Link-No with padding (padding is not shown) for a link to leaf occupy 1 byte; four fields L, Link-No, N-Branches and S-Branches+ for a link to a transit node occupy 2 bytes. In this case, the S-Branches+ field for link from PE1 to P1 is 14 (bytes), which is the size of the fields for 11 links (i.e., link from P1 to P2, link from P1 to P3, ..., link from P4 to PE7) encoding the sub-trees from P1 to PE2, PE3, ..., PE9. The S-Branches+ field for link from P1 to P2 is 8 (bytes), which is the size of the fields for 7 links (i.e., link from P2 to PE2, link from P2 to PE3, link from P3 to P4, and 4 links from P4 to PE4 - PE7) encoding the sub-trees from P1 to PE2 and PE3, and the fields following them.

Encoding the tree without L flag occupies 24 bytes in total as illustrated in Figure 3. Encoding the tree with L flag occupies 16 bytes in total as illustrated in Figure 4. 8 bytes (or say 33% of space/memory) are saved/reduced.

Figure 5 shows an enhancement of encoding the branch/sub-tree from P3 to PE4, PE5, PE6 and PE7 in Figure 1 using B flag. The B flag added for a link indicates whether the link bits+ are used to represent the link numbers and other link related information such as whether links’ remote end nodes (i.e., next hops) are leaves. When B flag for the link from upstream node such as P3 to downstream node such as P4 is set to one indicating the link bits+ are used, the links from downstream node such as P4 on the tree are encoded or represented by link bits+. A link bits+ contains a P (short for plus leaf) field, S-Bits (short for size of the bits in a unit such as byte) field, and Bits field.

P field has a value such as 1 indicating that a bit with value of 1 (one) in Bits field means that the corresponding link is on the branch and the link’s next hop is a leaf. There are four links to leaves (i.e., egress nodes) from P4, which are link from P4 to PE4 with link number 2, link from P4 to PE5 with link number 3, link from P4 to PE6 with link number 4, and link from P4 to PE7 with link number 5. These four links are represented by the 2-th bit, 3-th bit, 4-th bit and 5-th bit in the Bits field (from left to right) respectively.

S-Bits field has a value such as 1 indicating the size of the Bits field in a unit such as byte.

Suppose that the fields (i.e., L, B, Link-No, No-Branches and S-Branches+) for the link from P3 to P4 occupy 2 bytes; P and S-Bits occupy 1 byte. In this case, the link bits+ for the links from P4 occupy 2 bytes, encoding the branch from P3 uses 4 bytes in total.
The number of branches from P4 is the number of bits with value of one in the Bits field. The N-Branches field for the link from P3 to P4 is used for other purpose. For example, N-Branches field and S-Branches+ field combined to represent the size of the branches plus the fields following (i.e., S-Branches+).

<table>
<thead>
<tr>
<th>size</th>
<th>L</th>
<th>B</th>
<th>Link-No</th>
<th>N-Branches</th>
<th>S-Branches+</th>
<th>link</th>
<th>branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td>P3 to P4</td>
<td>branch</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td>0 1 1 1 0 0 0</td>
<td>P4 to PE4-PE7</td>
<td>to PE4-PE7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5: Encoding of branch from P3 via P4 to PE4, PE5, PE6, PE7 with B flag

Figure 6 shows an enhancement of encoding the branch part from PE1 via P1 to P2, P3, PE8 and PE9 on the P2MP tree/path in Figure 1 using B flag. B flag for the link from upstream node PE1 to downstream node P1 is set to one indicating that the link bits+ are used to represent the information about the links on the tree from downstream node P1.

P field has a value such as 0 indicating that a bit with value of 1 (one) in Bits field means that the corresponding link from downstream node P1 is on the tree.

S-Bits field has a value such as 1 indicating the size of Bits field in a unit such as byte.

Bits field of one byte has the second (2-th) bit, third (3-th) bit, 4-th bit and 5-th bit set to one (from left to right), indicating four links from P1 on the tree: the link from P1 to P2 with link number 2, link from P1 to P3 with link number 3, link from P1 to PE8 with link number 4 and link from P1 to PE9 with link number 5. The fields for each of these links do not have Link-No field, which is called reduced fields.
<table>
<thead>
<tr>
<th>size</th>
<th>L</th>
<th>B</th>
<th>Link-No</th>
<th>N-Branches</th>
<th>S-Branches+</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
<td>11</td>
<td>P1 to P1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>branch part</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>from PE1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>P1 to P2,P3, PE8,PE9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>to P2,P3, PE8,PE9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S-Bits</td>
<td>Bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td></td>
<td>S-Bits</td>
<td>Bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: Encoding of branch part from PE1 via P1 to P2, P3, PE8, PE9 with B flag

The reduced fields (i.e., L = 0, B = 0, No-Branches = 2, S-Branches+ = 6) for the link from P1 to P2 indicates that the link’s next hop is not leaf, link bits+ is not used for the branches from P2, the number of branches from P2 is 2 and the size of the branches from P2 plus the fields following is 6.

The reduced fields (i.e., L = 0, B = 0, No-Branches = 1, S-Branches+ = 4) for the link from P1 to P3 indicates that the link’s next hop is not leaf, link bits+ is not used for the branches from P3, the number of branches from P3 is 1 and the size of the branches from P3 plus the fields following is 4.

The reduced fields (i.e., L = 1) for the link from P1 to PE8 indicates that the link’s next hop PE8 is a leaf.

The reduced fields (i.e., L = 1) for the link from P1 to PE9 indicates that the link’s next hop PE9 is a leaf.

Suppose the fields (i.e., L, B, Link-No, No-Branches and S-Branches+) for a link to a transit node occupy 2 bytes; P and S-Bits occupy 1 byte; the reduced fields (i.e., L, B, No-Branches and S-Branches+) for a link to a transit node occupy 11 bits, the reduced fields (i.e., L) for a link to a leaf occupy 1 bit. In this case the reduced fields for four links from P1 to P2, P3, PE8 and PE9 uses 3...
bytes (i.e., 24 bits). Encoding the branch part from PE1 via P1 to P2, P3, PE8 and PE9 uses 7 bytes.

Figure 7 shows an enhancement of encoding the P2MP path/tree in Figure 1 using both L and B flags.

The branch part from PE1 via P1 to P2, P3, PE8 and PE9 is the same as the one in Figure 6 and occupies 7 bytes.

The link from P2 to PE2 is represented by L = 1 and Link-No = 1. The link from P2 to PE3 is represented by L = 1 and Link-No = 2. These two links occupy 2 bytes. In an option, when L = 1, the Link-No field and Pad are combined to represent link number.

The branch from P3 is the same as the one in Figure 5 and occupies 4 bytes. Encoding the tree of 13 nodes uses 13 bytes in total.

```
+=+=+===================================+ link
size |L|B|Link-No|N-Branches|S-Branches+|  link
+=+=+===================================+=======+====+==
13  |0|1|  2   |          |    11     | PE1 to P1
   |   |   |       |           |          | sub-tree
   |   |   |       |           |          | from PE1
11  |0|  1 |       | 0 1 1 1 1 0 0 0| P1 to P2,P3,
   |   |   |       |           |     PE8,PE9| to P1 to
   |   |   |       |           |          | PE2-PE9
   |P|     S-Bits     |     Bits      |
   |L|B|N-Branches|S-Branches+|                     |
+=+=+==========+===========+                     |
9       |0|0|   2   |          |     6     | P1 to P2
   |   |   |       |           |          |
8       |0|0|   1   |          |     4     | P1 to P3
   |   |   |       |           |          |
7       |1|                          | P1 to PE8
   |   |   |       |           |          |
6       |1|   1|       | 0 1 1 1 0 0 0| P2 to PE2
   |1|   2|       | P2 to PE3
5       |0|1|   2   |          |     2     | P3 to P4
   |   |   |       |           |          |
4       |0|1|   1   | 0 1 1 1 1 0 0 0| P4 to PE4
   |   |   |       |           |          |
3       |1|             |     2     |
   |   |   |       |           |
2       |1|             | 0 1 1 1 1 0 0 0| PE7
```

Figure 7: Encoding of tree from PE1 to PE2 - PE9 with L and B
4. Multicast Routing Header (MRH)

Figure 8 shows a Multicast Routing Header (MRH) in an IPv6 packet. The IPv6 packet comprises an IPv6 header with a destination address (DA) and source address (SA) of IPv6, a routing header with Routing type (TBD) indicating MRH and an IP multicast datagram. The routing header is indicated by the Next Header in the IPv6 header.

```
+-----------------+------------------+------------------------+
|   IPv6 header:  |  Routing header: | IP multicast datagram  |
|                 |                  | (IP datagram header +  |
|  Next Header =  | Next Header      |  data)                 |
|  Routing header |                  |                        |
| SA=IPv6 Address |Routing Type =    |                        |
| DA=IPv6 Address |   TBD (MRH)      |                        |
|                 | SL, b, nB        |                        |
|                 | sub-tree         |                        |
+-----------------+------------------+------------------------+
|<----   MRH  ---->|
```

Figure 8: Multicast Routing Header (MRH) in IPv6 packet

The format of the MRH is shown in Figure 9.

```
0                   1                   2                   3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header   |  Hdr Ext Len  |RoutingType=TBD|Version|Flags|b|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SL (Subtree Left) |nB (# Branches)|        Reserved         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Sub-tree encoded by link numbers                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Format of Multicast Routing Header (MRH)

The MRH has the following fields:

Next Header: The type of the header after the MRH. Either another extension header or the type of IP multicast datagram in the packet.

Hdr Ext Len: Its value indicates the length of the MRH in a unit of 64 bits (i.e., 8 bytes) excluding the first 8 bytes.
Routing Type: Its value TBD indicates that the routing header is a Multicast Routing Header (MRH) for TE multicast.

Version: The Version of the MRH. This document specifies Version zero.

Flags: A 1-bit b flag is defined. b = 1 indicates the links directly from the root of the sub-tree are encoded by bits+.

Sub-tree Left (SL): Its value as a pointer points to the sub-tree.

Number of Branches (nB): Its value indicates the number of branches from the root of the sub-tree.

Sub-tree: Its value encodes the TE multicast sub-tree using link numbers.

The P2MP path/tree from PE1 via P1 to PE2 - PE9 in Figure 1 is encoded by the link numbers of the links on the tree as illustrated in Figure 7. The link from ingress PE1 to P1 is encoded by the link number of the link on PE1, which is 2, the number of branches from P1 on the tree, and the size of the branches+ from P1, which is 11. The number of branches from P1 is the number of bits with value 1 in the Bits field for the links from P1 to P2, P3, PE8 and PE9, which is 4 since there are 4 bits with value 1 in the Bits field.

For an IP multicast packet to be transmitted by the P2MP path/tree, PE1 constructs an IPv6 packet for each sub-tree/branch from PE1 and sends the packet containing a MRH and the IP multicast packet to the next hop along the sub-tree.

The P2MP path/tree has one sub-tree from PE1 via P1 to PE2 - PE9. PE1 finds P1's IPv6 address from PE1's neighbor IPv6 address table using the link number 2 and sets DA (destination address) of the IPv6 packet to P1's IPv6 address and SA (source address) of the IPv6 packet to PE1's IPv6 address. PE1 builds the MRH based on the encoding of the tree in Figure 7 through setting b flag to 1, which is the value of B for the link from PE1 to P1, SL to 11, which is the value of S-Branches+ for the link from PE1 to P1. nB is not set since B = 1. Figure 10 shows the IPv6 packet to be sent to P1, which is received by P1. Note that some details are not shown.
The number of branches from P1 is 4, which is the number of bits with value 1 in the Bits field for the links from P1 to P2, P3, PE8 and PE9. This is determined by b = 1 and the Bits+ fields pointed by SL = 11.

There are 4 links from P1. The 2-th bit in the Bits field has value 1 indicating the link with link number 2, which is the link from P1 to P2. The reduced fields (i.e., L = 0, B = 0, No-Branches = 2, S-Branches+ = 6) for the link indicates that the link’s next hop P2 is not leaf, link bits+ is not used for the branches from P2, the number of branches from P2 is 2 and the size of the branches from P2 plus the fields following is 6.

The 3-th bit has value 1 indicating the link with link number 3, which is the link from P1 to P3. The reduced fields (i.e., L = 0, B = 0, No-Branches = 1, S-Branches+ = 4) for the link indicates that the link’s next hop P3 is not leaf, link bits+ is not used for the branches from P3, the number of branches from P3 is 1 and the size of the branches from P3 plus the fields following is 4.
The 4-th bit has value 1 indicating the link with link number 4, which is the link from P1 to PE8. PE8 is a leaf, which is indicated by the first L = 1. The 5-th bit has value 1 indicating the link with link number 5, which is the link from P1 to PE9. PE9 is a leaf indicated by the second L = 1.

The link number of the link from P2 to leaf PE2 is 1 on P2. The link number of the link from P2 to leaf PE3 is 2 on P2. The link number of the link from P3 to P4 is 2 on P3. The size of branches from P4 is 2. The link numbers of the links from P4 to leaf PE4, PE5, PE6 and PE7 are 2, 3, 4 and 5 on P4 respectively.

After receiving the IPv6 packet from PE1, P1 determines whether the packet’s next header is a MRH through checking if the next header is routing header and routing type in the routing header is TBD for MRH. When the next header is the MRH, P1 duplicates the packet for each link/branch/sub-tree from P1 and sends the packet copy with an updated MRH (note: only SL, b and nB are updated) to the next hop along the branch. Since b = 1, the number of bits with value 1 in the Bits field of Bits+ pointed by SL = 11 is the number of branches/links from P1 according to Figure 10, which is 4.

The 2-th bit in the Bits field has value 1 indicating the link with link number 2, which is the link from P1 to P2. The first reduced fields is for link from P1 to P2 and indicates the number of branches from P2 is 2 and the size of branches from P2 plus the ones following them is 6.

P1 duplicates the packet for the link with link number 2, finds P2’s IPv6 address from P1’s neighbor IPv6 address table using the link number 2 of the link from P1 to P2 and sets DA of the packet to P2’s IPv6 address. P1 updates the MRH based on the encoding of the sub-tree in Figure 10 through setting SL, b and nB to S-Branches+ = 6, B = 0 and N-Branches = 2 for the link from P1 to P2 respectively. Figure 11 shows the packet to be sent to P2, which is received by P2.
Figure 11: IPv6 packet with MRH received by P2

The number of branches from P2 is 2. The link number of the link from P2 to PE2 is 1 on P2. The link number of the link from P2 to PE3 is 2 on P2. PE2 and PE3 are leaves (i.e., egresses), which are indicated by L = 1.

The fields for encoding the branch/sub-tree from P3 follows the branches/sub-trees from P2.

The 3-th bit has value 1 indicating the link with link number 3, which is the link from P1 to P3. The second reduced fields is for link from P1 to P3 and indicates the number of branches from P3 is 1 and the size of branches from P3 plus the ones following them is 4.

P1 duplicates the packet for the link with link number 3 (which is link from P1 to P3) and sends the packet copy with an updated MRH to P3.

The 4-th bit has value 1 indicating the link with link number 4, which is the link from P1 to PE8. PE8 is a leaf, which is indicated by the first L = 1.

P1 duplicates the packet for the link with link number 4 and sends the packet copy with an updated MRH to PE8.

The 5-th bit has value 1 indicating the link with link number 5, which is the link from P1 to PE9. PE9 is a leaf, which is indicated by the second L = 1.
P1 duplicates the packet for the link with link number 5 and sends the packet copy with an updated MRH to PE9.

P1 duplicates the packet for the link with link number 3 (which is link from P1 to P3), finds P3’s IPv6 address from P1’s neighbor IPv6 address table using the link number 3 of the link from P1 to P3 and sets DA of the packet to P3’s IPv6 address. P1 updates the MRH based on the encoding of the sub-tree in Figure 10 through setting SL, b and nB to S-Branches+ = 4, B = 0 and N-Branches = 1 in the reduced fields for the link from P1 to P3 respectively. Figure 12 shows the packet to be sent to P3, which is received by P3.

```
| IPv6 Header | <-------- MRH --------> |
+-------------+-------------------------------+-------------+
| DA=P3’s IPv6|Routing Type=TBD, SL=4,b=0,nB=1|IP multicast |
| SA=PE1’s IPv6|sub-tree from P3 to PE4-PE7 |datagram     |
+-------------+-------------------------------+-------------+
```

<table>
<thead>
<tr>
<th>Link-No</th>
<th>N-Branch</th>
<th>S-Branches+</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 12: IPv6 packet with MRH received by P3

The number of branches from P3 is 1. The link number of the link from P3 to P4 is 2 on P3. The size of branches from P4 is 2. B = 1 for the link from P3 to P4 indicates that the links from P4 are encoded by link bits+. The number of branches from P4 is the number of bits with value 1 in the Bits field for the links from P4 to PE4 - PE7, which is 4 since there are 4 bits with value 1 in the Bits field.

After receiving the IPv6 packet from P1, P3 determines whether the packet’s next header is a MRH. When the packet’s next header is the MRH, P3 duplicates the packet for each link/branch/sub-tree from P3 and sends the packet copy with an updated MRH to the next hop along the branch. There is one branch from P3 according to the sub-tree remaining in the MRH in the packet received by P3. The branch/sub-tree is from the link from P3 to P4 towards PE4 - PE7.

P3 duplicates the packet for the branch/sub-tree, finds P4’s IPv6 address from P3’s neighbor IPv6 address table using the link number 2
of the link from P3 to P4 and sets DA of the packet to P4’s IPv6 address. P3 updates the MRH based on the encoding of the sub-tree in Figure 12 through setting SL and b to S-Branches+ = 2 and B = 1 for the link from P3 to P4 respectively. nB is not set or used since b = 1. Figure 13 shows the packet to be sent to P4, which is received by P4.

```
| IPv6 Header | <------- MRH -------> |
+-------------+-------------------------------+-------------+
|DA=P4’s IPv6 |Routing Type=TBD, SL=2,b=1,nB  |IP multicast |
|SA=PE1’s IPv6|sub-tree from P4 to PE4-PE7    |datagram     |
+-------------+-------------------------------+-------------+
```

![Figure 13: IPv6 packet with MRH received by P4](image)

After receiving the IPv6 packet from P3, P4 determines whether the packet’s next header is a MRH. When the packet’s next header is the MRH, P4 duplicates the packet for each branch/link from P4 and sends the packet copy with an updated MRH to the next hop along the branch. There are 4 branches/links from P4 according to the sub-tree remaining in the MRH in the packet received by P4.

The 2-th bit in the Bits field has value 1 indicating the link with link number 2, which is the link from P4 to leaf PE4.

The 3-th bit has value 1 indicating the link with link number 3, which is the link from P4 to leaf PE5.

The 4-th bit has value 1 indicating the link with link number 4, which is the link from P4 to leaf PE6.

The 5-th bit has value 1 indicating the link with link number 5, which is the link from P4 to leaf PE7.

P4 duplicates the packet for the first branch, finds PE4’s IPv6 address from P4’s neighbor IPv6 address table using the link number 2 of the link from P4 to PE4 and sets DA of the packet to PE4’s IPv6 address. P4 updates the MRH based on the encoding of the sub-tree in Figure 13 through setting SL to 0 since PE4 is a leaf. Figure 14 shows the packet to be sent to PE4, which is received by PE4. Similarly, P4 duplicates the packet for each of the other links, updates the MRH and sends the packet copy to each of PE5, PE6 and PE7.
After receiving the packet from P4, PE4 determines whether the packet’s next header is a MRH. When the packet’s next header is the MRH, PE4 checks if PE4 itself is a leaf (i.e., egress) through checking whether SL is 0. When PE4 is a leaf, PE4 decapsulates the packet and sends the IP multicast datagram to the IP multicast forwarding module.

Alternatively, after receiving the packet from P3 and determining that the packet’s next header is a MRH, P4 checks if each of its next hops is a leaf. When the next hop is a leaf, P4 decapsulates the packet and sends the IP multicast datagram to the next hop. Since 4 next hops PE4 - PE7 are leaves, P4 sends the IP multicast datagram to PE4 - PE7.

5. Procedures/Behaviors

This section describes the procedures or behaviors on the ingress, transit and egress/leaf node of a P2MP path to deliver a packet received from the path to its destinations.

5.1. Procedure/Behavior on Ingress Node

For a packet to be transported by a P2MP path, the ingress of the P2MP path duplicates the packet for each sub-tree/branch of the P2MP path branching from the ingress, encapsulates the packet copy in a MRH containing the sub-tree and sends the encapsulated packet copy to the next hop node along the sub-tree.

For example, there is one sub-tree branching from the ingress of the P2MP path/tree in Figure 1. The sub-tree is from ingress PE1 via next hop node P1 towards PE2 to PE9. PE1 sends P1 the packet as shown in Figure 10.

5.2. Procedure/Behavior on Transit Node

When a transit node of a P2MP path/tree receives a packet transported by the P2MP path/tree, the node determines whether the current routing header is a MRH. After determining that it is a MRH, the node executes the procedure to duplicate the packet for each of the
downstream links of the node on the P2MP path/tree and send the packet copy to next hop (i.e., downstream node) of the link.

Suppose that the transit node receives the packet from a upstream link from a upstream node to the transit node and there are n downstream links from the transit node on the P2MP path/tree (i.e., there are n branches/sub-trees from the transit node on the P2MP path/tree, where n is greater than zero).

The information about the upstream link is in b and nB field of the MRH. When b = 0, nB field contains the number of branches from the transit node. The information about n downstream links is pointed by SL. When b = 1, the number of branches from the transit node is the number of bits with value 1 in the Bits field of the Bits+ fields pointed by SL. The information about n downstream links is encoded by the Bits field and the fields following the Bits field.

For example, when node P1 receives the packet transported by the P2MP path/tree in Figure 1 from ingress PE1, the MRH is illustrated in Figure 10. The b = 1, the Bits field of the Bits+ fields pointed by SL = 11 has 4 bits with value 1. So there are 4 branches from P1. The information about 4 downstream links (i.e., link from P1 to P2, link from P1 to P3, link from P1 to PE8 and link from P1 to PE9) is encoded by the Bits field and the reduced fields for these 4 links following the Bits field.

For each of the downstream links of the transit node, the transit node duplicates the packet for the link, sets SL, b and nB in the packet copy accordingly. If the next hop is a leaf (i.e., egress), the transit node sets SL to 0; otherwise, the transit node sets SL, b and nB to the value of S-Branches+, B and N-Branches for the link respectively when B is 0. When B = 1, the transit node sets SL and b to the value of S-Branches+ and B for the link respectively. The transit node finds the IPv6 address of the next hop (i.e., the downstream node) of the link from the neighbor IPv6 address table of the transit node using the link number of the link, sets the DA of the packet copy to the IPv6 address of the next hop, and sends the packet copy to the next hop of the link.

For example, for the first downstream link from P1 (i.e., link from P1 to P2), P1 duplicates the packet for the link, sets SL to 6 (which is the value of the S-Branches+ field for the link), b to 0 (which is the value of B for the link) and nB to 2 (which is the value of the N-Branches field for the link). P1 finds the IPv6 address of the next hop P2 of the link from the neighbor IPv6 address table of P1 using the link number 2 of the link, sets the DA of the packet copy to the P2’s IPv6 address, and sends the packet copy to P2. The packet copy received by P2 is shown in Figure 11.
For the second downstream link from P1 (i.e., link from P1 to P3), P1 duplicates the packet for the link, sets SL, B, and nB to 4, 0, and 1 respectively, which are the values of S-Branches+, B, and N-Branches for the link from P1 to P3 respectively. P1 finds the IPv6 address of the next hop P3 of the link from the neighbor IPv6 address table of P1 using the link number 3 of the link, sets the DA of the packet copy to the P3’s IPv6 address, and sends the packet copy to P3. The packet copy received by P3 is illustrated in Figure 12.

For the 3-th downstream link from P1 (i.e., link from P1 to PE8), P1 duplicates the packet for the link, sets SL to 0 since PE8 is a leaf (i.e., egress). P1 finds the IPv6 address of the next hop PE8 of the link from the neighbor IPv6 address table of P1 using the link number 4 of the link, sets the DA of the packet copy to the PE8’s IPv6 address, and sends the packet copy to PE8.

5.3. Procedure/Behavior on Egress Node

When an egress node of a P2MP path receives a packet transported by the path, the DA of the packet is the IPv6 address of the egress node and there is an indication in the MRH for the leaf/egress. The egress node proceeds to process the next header in the packet.

For example, after receiving the IPv6 packet from P4, PE4 determines whether the packet’s next header is a MRH. When the packet’s next header is the MRH, PE4 checks if PE4 itself is a leaf (i.e., egress) through checking whether SL is 0. When PE4 is a leaf, PE4 decapsulates the packet and sends the IP multicast datagram to the IP multicast forwarding module.

6. IANA Considerations

This document requests assigning a new Routing Type in the subregistry "Routing Types" under registry "Internet Protocol Version 6 (IPv6) Parameters" as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD (7 suggested)</td>
<td>Multicast Routing Header</td>
<td>This document</td>
</tr>
</tbody>
</table>

7. Security Considerations

TBD
8. Acknowledgements

The authors would like to thank Donald Eastlake for the valuable comments and suggestions on this draft.

9. References

9.1. Normative References


9.2. Informative References

[I-D.chen-pim-srv6-p2mp-path]

[I-D.ietf-pim-sr-p2mp-policy]

Authors’ Addresses

Huaimo Chen
Futurewei
Boston, MA
USA

Email: Huaimo.chen@futurewei.com
Mike McBride
Futurewei

Email: michael.mcbride@futurewei.com

Yanhe Fan
Casa Systems
USA

Email: yfan@casa-systems.com

Zhenbin Li
Huawei

Email: lizhenbin@huawei.com

Xuesong Geng
Huawei

Email: gengxuesong@huawei.com

Mehmet Toy
Verizon
USA

Email: mehmet.toy@verizon.com

Gyan S. Mishra
Verizon
13101 Columbia Pike
Silver Spring MD 20904
USA

Phone: 301 502-1347
Email: gyan.s.mishra@verizon.com

Yisong Liu
China Mobile

Email: liuyisong@chinamobile.com
Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing  102209
China
Email: wangaj3@chinatelecom.cn

Lei Liu
Fujitsu
USA
Email: liulei.kddi@gmail.com

Xufeng Liu
IBM Corporation
USA
Email: xufeng.liu.ietf@gmail.com
Encoding Network Slice Identification for SRv6
draft-cheng-spring-srv6-encoding-network-sliceid-04

Abstract

This document describe a method to encode network slicing identifier within SRv6 domain.

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 08, 2023.

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.
1. Introduction

SRv6 Network Programming[RFC8986] enables the creation of overlays with underlay optimization to be deployed in an SR domain[RFC8402].

As defined in [RFC8754], all inter-domain packets are encapsulated for the part of the packet journey that is within the SR domain. The outer IPv6 header [RFC8200] is originated by a node of the SR domain and is destined to a node of the SR domain.

This document describes a novel method to encode slice identifier in the outer IPv6 header of an SR domain. Unlike other proposed methods before, which will bring side effects on existed functions, by encoding network slicing identifier in the source IPv6 address of the outer header, this method avoids the drawbacks which previous proposals incur.

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. Slice Identifier

Slice identifier (SLID) is a 16-bit Identifier which uniquely defines a slicing of the network in the specified SR domain.

3. SLID Assignment

When SR domain enables network slicing, the ingress PE should reserve 16 least significant bits in its locater for slicing use. When a packet enters the SR domain from an ingress PE, the ingress PE encapsulates the packet in an outer IPv6 header and optional SRH as defined in [RFC8754]. The ingress PE MAY also classify the packet into a slice and set the slice identifier as follows:
- Set the SPI bit (SLID Presence Indicator) in the Traffic Class field of the outer IPv6 header.
- Write this SLID in the 16 least significant bits of source address of the outer IPv6 header.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|   SPI Bit     |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                         Source Address                        |
|                                                               |
|                                                               |
|                                                               |
|                           +----------------------------------+
|                           |                   SLID                  |
|                           +----------------------------------+
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                      Destination Address                       |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                                 |
|                                                                 |
|                                                                 |
|                                                                 |
|                                                                 |
|                                                                 |
|                                                                 |
|                                                                 |
|                                                                 |
|                                                                 |
|                                                                 |
|                      +----------------------------------+
|                      |                   Hop Limit            |
|                      +----------------------------------+
```

The choice of the SPI bit from within the IPv6 Traffic Class field is a domain-wide configuration and is outside the scope of this document.

4. Per-Slice Forwarding

Any router within the SR domain that forwards a packet with SPI bit set uses the SLID to select a slice and apply per-slice policies.

5. Backward Compatibility

PE routers that do not set the SPI bit do not enable the SLID semantic of the IPv6 source address bits. Hence, SLID-aware routers would not attempt to classify these packets into a slice.

Any router that does not process the SPI nor the SLID forwards packets as usual.
6. Acknowledgements

The authors would like to thank AAAA, BBBB and CCCC for their insightful feedback on this document.

7. References

7.1. Normative References


7.2. Informative References


Authors’ Addresses

Weiqiang Cheng  
China Mobile  
Beijing  
China  
Email: chengweiqiang@chinamobile.com

Changwang Lin  
H3C Corporation  
Beijing  
China  
Email: linchangwang.04414@h3c.com

Liyan Gong  
China Mobile  
Beijing  
China  
Email: gongliyan@chinamobile.com

Shay Zadok  
Broadcom
Israel
Email: shay.zadok@broadcom.com

Xuewei Wang
Ruijie Networks Co., Ltd.
Beijing
China
Email: wangxuewei1@ruijie.com.cn

Cryptographically Generated Addresses (CGA) Light
draft-ev-6man-cga-light-00

Abstract

This document specifies a method for securely associating link-layer addresses (MAC) to the IPv6 address by a cryptography method similar to blockchain mining. It permits guarantee security at the IPv6 layer to the same degree as at the link layer (that node is dependent on anyway).

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 2021.

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 (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. Terminology and pre-requisite........................................2
2. Introduction..........................................................3
3. Strength Evaluation...................................................5
4. IID generation technical details.................................6
5. IID validation technical details....................................9
6. ND extensions to exchange relevant information...............11
7. Compatibility with other ND functionality......................12
8. Security Considerations............................................13
9. IANA Considerations................................................14
10. References..........................................................14
    10.1. Normative References.....................................14
    10.2. Informative References...................................15
11. Acknowledgments..................................................16

1. Terminology and pre-requisite

Many terms are inherited from [CGA] and [SEND].

Information for IID - the collection of information needed to distinguish IIDs generated for different purposes in compliance with other standards; local information to the node.

Digest of IID information - hash from "Information for IID"; public information distributed by ND.

Chg parameter - a number between 0 and 15 (4 bits) that participates in the calculation (A+B*Chg) of leading zero bits that should be in a validated hash; "A" and "B" are constants fixed in this specification

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

Expires January 8, 2023
2. Introduction

There are many attack vectors to ND. [ND Trust Model] section 4.1 has the name "Non-router related threats" and could be summarized:

- A malicious node could answer DAD for any request of a legitimate node (denial of service attack)
- A malicious node could poison the cache of another node (especially the router) to intercept traffic directed to another node (man in the middle attack); it is possible for neighbor solicitation and neighbor advertisement in many different cases.

The potential bunch of attacks based on the above is discussed in detail in [ND MITM] section 3:

- Rewrite cache by unsolicited NA
- Be the first and suppress DAD
- Win the race just after DAD

The [SEND] plus [CGA] have proposed cryptographic solutions to the problem. The [CGA] was positioned as dependent on [SEND] not a separate solution, it was used primarily to avoid unnecessary signature verifications (pre-check) for denial of service requests. Unfortunately, [SEND] has a pre-request to organize PKI (public key infrastructure) which is a well-known problem for enterprises. It is assumed that trusted anchor management in the [SEND] solution was the primary reason for low adoption. Effectively, it could be said that [SEND] has low adoption for the same reason as IPSec: a challenge to organize key management. Blockchain’s general success has shown that cryptography could be valuable even in a situation when the trusted anchor is not considered.

This document is based on the [CGA] with minor modifications needed to satisfy the requirements of many other standards developed since then. This document fulfills the [CGA] promise: "The protection works without a certification authority or any security infrastructure".

The [CGA] was associating the "public key" to "IID" as a backup to public key management in the certification authority.
In contrast, this document proposes a "Link-layer" to "IPv6" addresses associated with cryptography assurance.

Link-layer to IP layer mapping is the primary function of [ND] protocol that is protected by this specification.

It is possible to have the link with nodes supporting this specification and legacy nodes that could not check the address ownership. ND cache could be poisoned on unprotected nodes in a mixed environment. The router is the key target for ND cache poisoning. Hence, router support for this specification is especially valuable.

A reminder that hash function is by design has no correlation between output and input. Hence, only brute input trials are needed to find output with any desired property (like some number of leading zeroes).

The protection principles here are the same as for blockchain mining or [CGA]: the legal node needs $2^{(A+B*Chg+1-1)}$ calculations on average for IID mining but the malicious node would need $2^{(A+B*Chg+60-1)}$ calculations on average because in addition to the same amount of leading zeroes it needs additionally the particular combination of 60 trailing bits at the same time. "+1" in the above calculations is because the hash is double for mining. "-1" in the above calculation is because of the average (50% probability).

Hence, the intruder needs $2^{59}$ times more hash calculations to find a proper "Digest of IID information" that would match his link-layer address (MAC).

[Hash in Sec] has a discussion that if many hash algorithms are available on the link then intruder can attack the weakest. [Hash in Sec] has proposed to encode hash type in the IID (borrowing security level space), IANA "CGA SEC" registry has been created. Hence, [CGA] security level ("Sec" parameter) has a different meaning from the "Chg" parameter used in this specification.

Attack vector on hash type downgrade has been dealt with in this specification by rewrite rules of ND cache where hash type is considered.

Let’s overview IID mining:

0. Choose the Chg parameter that points to how many leading zeroes should be in the hash on step 3.
1. Initially, all relevant information for IID creation is concatenated in the clear text block that would be named
"Information for IID". It is primarily the information needed for:
   a. Guaranty IID uniqueness (Time, Nonce, Collision count)
   b. Guaranty stable but different IID for different link attachments (interface name, network name)
2. Then "Information for IID" is hashed. The result is the "Digest of IID information" that would be made public by [ND] protocol.
3. "Digest of IID information" is concatenated with Prefix and link-layer (MAC) and then hashed again.
4. If the hash result of step 3 already has the needed number of leading zeroes then 60 lowest bits are used for IID and "Information for IID" should be saved in non-volatile memory. Else (not enough leading zeroes) any bits (Nonce, Time) or fields order in the "Information for IID" should be changed and this algorithm loops to step 2.

The algorithm convergence may need $2^{(A+B*Chg)}$ calculations in an average of the selected hash function, it has a good probability to converge for $2^{(A+B*Chg+1)}$ hash calculations.

Any other node may do a simple check of validity for the Prefix/IID/Link-Layer combination before rewriting ND cache:

0. Extract Chg from 4 high order IID bits.
1. Hash the "Digest of IID information" concatenated with Prefix and link-layer address (MAC).
2. Check that the leading $A+B*Chg$ bits of the hash are zeroes. Discard ND update otherwise.
3. Check that the trailing 60 bits of the hash are equal to the IID claimed by the node. Discard ND update otherwise.
4. Accept ND update.

3. Strength Evaluation

Bitcoin is a good reference for such a type of evaluation. It uses a double hash like this specification, the type is SHA-256. The different hash algorithms (SHA-512 is recommended below for this specification) may improve security that would be ignored for this estimation.

Bitcoin has started from the requirement for 32 bits leading zeroes (for the genesis block) and has moved to 80 bits now. It has surpassed 200EH/s ($2*10^{20}$ hashes per second) to crack 80 bits for 10 minutes on average.

The principal difference is that the Bitcoin block (to hash) is much bigger (around 1MB+-50%) compare needed to CGA (around 280B). Hence,
Bitcoin’s difficulty is around $2^{12}$ bigger just because of the bigger input.

Let’s justify the "A" and "B" parameters for "A+B*Chg" leading zeroes.

"A" leading zeroes should be easy to calculate even for the smallest system. The smallest IoT system may spend as high as 200us for one SHA-256 in software (hardware acceleration is available very often then the performance is better). 1 minute is the assumption for the time permitted to spend for mining then A=8 leading zeroes that are needed for mining at a minimum.

The challenge for the intruder would be $2^{67}$ hashes. The whole Bitcoin infrastructure would crack it for 18us. It would be just a problem to persuade mining owners to proceed.

Let "B" would be big enough for the whole Bitcoin infrastructure would not crack IID in 1000 years. 1000 years is different from 10 min in $2^{26}$. Additionally, our hashed block is $2^{12}$ smaller. Hence, it should be an 80+26+12=118 bits challenge. Then B=4.

The challenge for mining IID would be $2^{68}$. Some GPUs are capable of 3GH/s then IID mining would be around 3000 years on one GPU which is probably a good room for growth. It means that the Chg parameter at the highest values (>12) would be seen as very rare for some time.

Summary: for respective A+B*Chg, A=8, B=4.

Chg is transmitted in every ND update.

It is the compromise between the possibility of mining on the smallest node and maximum protection that is desired to achieve in the future for a very important server.

In general, knowing the node capability for the number of hashes per second (for chosen hash type) - it is easy to predict the average time of mining for the specific level of protection (Chg parameter).

4. IID generation technical details

IID generation should be done separately per every link/network according to [Different IIDs]. It would be needed to perform this procedure after any new prefix (PIO) announcement or link-layer address change.

The implementation MAY have the capability to get initial parameters from the configuration when it is possible to do mining on a different system. In that case, the target system SHOULD go through the algorithm below anyway to check the consistency. It would cost 2 hash calculations to check that all data are consistent.
Decide about the needed level of protection for the address. Chg parameter has 4 bits that permit 16 levels of protection, every next level of protection would request "B" additional zero leading bits in the second hash mining. All nodes SHOULD have a possibility of administrator configuration for the level of IID protection (Chg parameter). The default protection level for the particular system is up to the implementer.

Like all other cases in cryptography, even well-protected IID (high levels of Chg) MUST be refreshed after some time. Pay attention that Chg increase by one would increase the strength of IID by $2^B$. It does not permit giving default configuration recommendations because different hash algorithms by themselves may have different strengths and different target systems may have a few orders of magnitude different attractiveness for an intruder that may bring principally different resources for mining. Hence, the table of lifetimes for every level of Chg is implementation-specific. It may be different for different hash algorithms.

Reminder, that [IID significance] has deprecated special meaning for "u" and "g" bits in IID. Hence, all 64 bits are available for our specification. 4 bits were used for the protection level. Hence, 60 bits are left for random ID.

Step 1:

As discussed in [Temporary addresses] that "Information for IID" should have a prefix, link-layer address, network name, time, DAD counter, and some nonce to make it possible for temporary address generation. Prefix and link-layer addresses would be added to the second hash calculation. Hence, initially, the "Information for IID" text should have:

- Nonce to easy change the "Information for IID" hash
- Time in any format, better to follow [Temporary addresses] assumptions
- Network name (like SSID) for links that have such parameter
- DAD collision count that is regulated by other specifications
- Any other parameters are permitted because all parameters of "Information for IID" have local meaning - they would be not advertised by [ND]

The size of the parameters is up to the implementer. An especially big block of "Information for IID" may affect the performance of IID mining.
The order of information concatenation is not important. Moreover, as it would be shown later the order may be changed during the mining process.

Special warning about Nonce size. It is RECOMMENDED to choose it longer than chosen A+B*Chg bits. Or else would be the same problem as for Bitcoin: Nonce could be looped around, but mining is not successful yet. In this case, like for Bitcoin, it is possible to change parameters (for example, read new time) or reorder the above-mentioned fields or introduce extraNonce as an additional field to the "Information for IID". To avoid this additional algorithm complexity, it is better to define Nonce as A+B*15 bits which should be enough for the most secure IID mining. Anyway, it is a local implementation matter.

It does not make sense to spend resources on additional mathematical functions upon "Information for IID" because later it would be hashed anyway for good randomization. The hash function is good enough to satisfy [Temporary addresses] requirements.

Step 2:

Then "Information for IID" is hashed. The result is the "Digest of IID information" that would be made public by [ND] protocol together with the corresponding link-layer address and prefix intended for IID.

The hash at this stage has local significance. Hence, the hash type may be different from the hash type in Step 3 which should be synchronized between nodes on the link. Simplicity should have both hashes of the same type.

Step 3:

"Digest of IID information" is concatenated with Prefix and link-layer address into one block that is hashed again.

It is MANDATORY to keep the order and size mentioned in this Step parameters to have consistent hash results on all nodes of the link (to check validity).

"Digest of IID information" on the input MUST have the same size as the output of the chosen hash type for this step. If there is any difference then Digest should be truncated or fitted with leading zeroes. Most probably, both steps (step 2 and step 3) would use the same hash type then the hash size would be equal automatically.
The prefix MUST be populated with trailing zeroes up to 128 bits to support any prefix boundary in the future.

Link Layer address MUST be fitted leading zeroes to 64 bits to be capable to accommodate any L2 technology.

Step 4:

Check for how many leading zeroes the second hash has (from the previous step). Mining SHOULD be processed till A+B*Chg leading bits are zeroed. If successful then the resulted "Information for IID" and "Digest of IID information" SHOULD be memorized in non-volatile memory. Else "Information for IID" should be modified and the mining algorithm should be looped to step 2.

"Information for IID" has many ways to be modified:

. Increment Nonce
. Reorder fields in the "Information for IID"
. Change fields (update time)
. Add field (like extraNonce)

It is potentially possible in Step 4 to memorize the "Information for IID" for the best result seen. Then it is possible to break from the mining with the correction of the Chg to the achieved number. Hence, it is possible to loop in the mining cycle for a limited time. It is NOT RECOMMENDED because it does not permit achieving the desired level of security. But it may be needed for the high protection levels (high Chg) or on systems with low hash processing rates to avoid the system hanging in the mining for an unacceptable period.

Step 5:

According to the [SLAAC] procedure, the new IID should be checked for uniqueness by DAD. If DAD is positive then Collision Counter in the "Information for IID" is incremented and the mining is looped to step 2. It is up to other specifications to regulate how many collisions are tolerable and what to do after it would be reached.

5. IID validation technical details

The validation node SHOULD check the IP/Link-layer pair association before any modification in the ND cache. And even after the validity
is checked, there is an additional procedure to check overwrite rights (see at the end of this section).

Every record in the ND cache should have a flag "validated" and additional fields used for validation (at least "Digest of IID information" and hash type). Let’s call legacy ND records and updates as Not-Protected.

The validating node should receive an ND message with all needed information (hash algorithm type, "Digest of IID information", Prefix, link-layer address) by mechanisms specified in section 6. Partial presence of needed parameters (for example "Digest of IID information" is present but the hash type is absent) is treated as an update request from a node not supporting this specification (Not-Protected node).

The mining challenge parameter (Chg) should be extracted from 4 high-order bits of IID.

See details in section 4. Step 3 for the importance of the order and size of all concatenated fields to get consistent results. Then hash the "Digest of IID information" concatenated with Prefix and link-layer address. Resulted hash should be checked for:

- leading A+B*Chg bits of the hash are zero
- trailing 60 bits of the hash are equal to the IID claimed by the node

If any check is negative then the ND update MUST be dropped. The update that claims to be protected but failed validation should not be considered in principle.

If both checks are positive then proceed to the final check that is especially needed in the mixed environment:

- The Not-Protected record SHOULD be overwritten by a Validated ND update.
- The Validated record SHOULD be overwritten by a Validated ND update that satisfies two checks: 1) equal or high permitted (see section 6. about the possibility to exclude some hash types from consideration) hash type number and 2) high Chg parameter (more leading zeroes for protection).
- The Validated record MUST NOT be overwritten by 1) a Not-Protected ND update or 2) by a Validated update with an equal or lower protection level (the smaller or equal Chg parameter) or 3) by a Validated update with lower permitted (see section
6. about the possibility to exclude some has types from consideration) hash type number. Effectively, the last rule may be presented in a simpler form: The Validated record MUST NOT be overwritten in all other cases.

6. ND extensions to exchange relevant information

For the other node to check address ownership, it needs "Digest of IID information" concatenated with Prefix and link-layer address. 3 parameters and the type of the hash function should be advertised.

ND option 39 (Crypto-ID Parameters) would be reused for the hash type signaling. All other parameters delivered by this option except hash function type would be ignored for this specification. Crypto-Type number 1 is pointing to SHA-512 which is RECOMMENDED as the default. Crypto-type section of the [ICMPv6] registry currently has types 0 or 2 that choose SHA-256. Crypto-type registry may be extended as needed. Different nodes can have different crypto-types on the link, hence, nodes (especially routers) may need to support all active hash types on the link.

Nodes MUST have the capability to restrict the list of hash types accepted on the link. It is needed to block failed or weak hash algorithms in the future. The implementation MAY just treat such updates as Not-Protected without checking the validity and following rewrite rules for Not-Protected updates. It is RECOMMENDED that by default node should accept in ND updates only the hash type that the node itself is using for mining, all other hash types are better to provision consciously.

It is the primary purpose of ND to deliver a link-layer address. [ND] protocol is already asking to include link-layer address options for all relevant cases:

- It should be included for Router Solicitation (except the IP address is still unspecified)
- It may be omitted for Router Advertisement
- It must be included for multicast and should be included for unicast of Neighbor Solicitation
- It must be included for multicast and should be included for unicast of Neighbor Advertisement (except IP address is still unspecified)

Hence, it may be assumed that the link-layer address would be available at the time needed.

Expires January 8, 2023
ND protocol is based on IP. Hence, the prefix could be extracted from the source IP address.

The only parameter left is the "Digest of IID information". It is defined in this specification:

```
+-----------------+-----------------+-----------------+-----------------+
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |             Digest            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                of IID information (hash)                      ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The type number is TBD by IANA.

"Digest of IID information" and "Crypto-type" options SHOULD be sent in all cases when link-layer information is sent, i.e. on all occasions when [ND] uses option 1 (source link-layer address) or option 2 (target link-layer address).

If the node does not support this specification or requested hash type then it could not protect other node address ownership. Hence, it is mandatory to support this specification and all needed hash types at least on routers.

7. Compatibility with other ND functionality

This specification applies to all types of IPv6 addresses: LLA, ULA, and GUA.

ND Proxy claims the ownership of link-layer addresses on different media. Hence, it would be operational for its primary purpose - to interconnect separate media. ND Proxy fully impersonalizes the IP/Link-layer relationship for the different media. ND Proxy does not need to do mining - it should reuse all parameters received from the original node owning the IP/Link-layer association.
This specification does not influence the protocol exchange. Hence, it is compatible with [ND] and its many modifications (Optimistic DAD, NUD improvements, Gratuitous ND, Multihoming, etc).

The proposal may replace deep snooping and automatic filtering by SAVI.

Many node redundancy schemes share link-layer addresses (MAC) then it is not a problem to share "Information for IID" between such nodes and keep IP to link-layer association protected.

The load balancer should have no problem distributing traffic between unique IP/Link-layer pairs.

There is a special situation with Anycast if it is localized on one link because the same IP could not have a different link-layer address. Hence, this specification should be disabled on the routers for this link.

Anycast distributed between links (a more typical situation) is not a problem because the IP to the link-layer association has a local significance for the link.

8. Security Considerations

This document closes some ND vulnerabilities by cryptographic methods popular in the blockchain. IP Address ownership is securely associated with the link-layer address.

This specification does not prevent replay attacks. If the victim node is absent on the link then the malicious node could change his link-layer address (MAC) to the victims' then claim IP/Link-layer association replaying previously saved ND messages. It may not happen for link layers that are protected by themselves (typically encryption implemented on any wireless). If successful, then the intruder may impersonate the absent victim and ask for credentials as a minimum. It has a low probability because the server is typically always available but the client is not receiving incoming connections. As it was stated in the abstract the level of protection is equal to the level of link layer protection. If it is possible to impersonate the address at the link-layer then IP layer protection would not help, certification authority and end-to-end traffic encryption are needed.

Replay protection needs different Nonce included into every message then encrypt and decrepit every message that is a very expensive proposition for [ND].
Canceling a trusted anchor would make all nodes equal that does not permit the restriction of the router's functionality to particular nodes. Hence, this specification could not deprecate [RA-Guard] and [RA-Guard+].

This specification does not prevent denial of service attacks in general. It is possible to generate many legal IP/Link-layer associations (presumably with a low level of mining by Chg parameter) and try to overwhelm other nodes on the link with additional states.

9. IANA Considerations

The new option "Digest of IID information Option" is specified with the type number TBD.

It is registered in the section "IPv6 Neighbor Discovery Option Formats" of the [ICMPv6] registry.

10. References

10.1. Normative References


10.2. Informative References


11. Acknowledgments

Thanks to the 6man working group for problem discussion.

Authors’ Addresses

Eduard Vasilenko
Huawei Technologies
17/4 Krylatskaya st, Moscow, Russia 121614
Email: vasilenko.eduard@huawei.com
Carrying Virtual Transport Network (VTN) Information in IPv6 Extension Header
draft-ietf-6man-enhanced-vpn-vtn-id-01

Abstract

Virtual Private Networks (VPNs) provide different customers with logically separated connectivity over a common network infrastructure. With the introduction and evolvement of 5G and other network scenarios, some existing or new customers may require connectivity services with advanced characteristics comparing to traditional VPNs. Such kind of network service is called enhanced VPNs (VPN+). VPN+ can be used to deliver IETF network slices, and could also be used for other application scenarios.

A Virtual Transport Network (VTN) is a virtual underlay network which consists of a set of dedicated or shared network resources allocated from the physical underlay network, and is associated with a customized logical network topology. VPN+ services can be delivered by mapping one or a group of overlay VPNs to the appropriate VTNs as the virtual underlay. In packet forwarding, some fields in the data packet needs to be used to identify the VTN the packet belongs to, so that the VTN-specific processing can be performed on each node the packet traverses.

This document proposes a new Hop-by-Hop option of IPv6 extension header to carry the VTN related information, which could used to identify the set of network resources allocated to a VTN and the rules for packet processing. The procedure for processing the VTN option is also specified.

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 12 January 2023.

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
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 .............................................  3
   1.1.  Requirements Language ...............................  4
2.  New IPv6 Extension Header Option for VTN ..................  4
3.  Procedures ...............................................  5
   3.1.  Adding VTN Option to Packet ........................  6
   3.2.  VTN based Packet Forwarding ........................  6
4.  Operational Considerations ...............................  7
5.  IANA Considerations .....................................  7
6.  Security Considerations ..................................  7
7.  Contributors .............................................  7
8.  Acknowledgements ........................................  8
9.  References ...............................................  8
   9.1.  Normative References ...............................  8
   9.2.  Informative References .............................  8
Authors’ Addresses .......................................... 10
1. Introduction

Virtual Private Networks (VPNs) provide different customers with logically isolated connectivity over a common network infrastructure. With the introduction and evolvement of 5G and other network scenarios, some existing or new customers may require connectivity services with advanced characteristics comparing to traditional VPNs, such as resource isolation from other services or guaranteed performance. Such kind of network service is called enhanced VPN (VPN+). VPN+ service requires the coordination and integration between the overlay VPNs and the capability and resources of the underlay network. VPN+ can be used to deliver IETF network slices [I-D.ietf-teas-ietf-network-slices].

[I-D.ietf-teas-enhanced-vpn] describes a framework and the candidate component technologies for providing VPN+ services. It also introduces the concept of Virtual Transport Network (VTN). A VTN is a virtual underlay network which consists of a set of dedicated or shared network resources allocated from the physical underlay network, and is associated with a logical network topology. VPN+ services can be delivered by mapping one or a group of overlay VPNs to the appropriate VTNs as the underlay, so as to provide the network characteristics required by the customers. In packet forwarding, traffic of different VPN+ services needs to be processed separately based on the network resources and the logical topology associated with the corresponding VTN. In the context of network slicing, VTN and NRP are considered as similar concepts, and NRP can be seen as an instantiation of VTN.

[I-D.dong-teas-nrp-scalability] describes the scalability considerations and the possible optimizations for providing a relatively large number of VTNs for VPN+ services. One approach to improve the data plane scalability of VTN is to introduce a dedicated VTN Resource Identifier (VTN Resource ID) in the data packet to identify the set of network resources allocated to a VTN, so that VTN-specific packet processing can be performed using that set of resources, which avoids the possible resource competition with services in other VTNs. This is called Resource Independent (RI) VTN. A VTN Resource ID represents a subset of the resources (e.g. bandwidth, buffer and queuing resources) allocated on a given set of links and nodes which constitute a logical network topology. The logical topology associated with a VTN could be defined using mechanisms such as Multi-Topology [RFC4915], [RFC5120] or Flex-Algo [I-D.ietf-lsr-flex-algo], etc.

This document proposes a mechanism to carry the VTN resource ID together with other VTN related information in a new Hop-by-Hop option called "VTN option" of IPv6 extension header [RFC8200] of IPv6.
packet, so that on each network node along the packet forwarding path, the VTN option in the packet is parsed, and the obtained VTN Resource ID is used to instruct the network node to use the set of network resources allocated to the corresponding VTN to process and forward the packet. The procedure for processing the VTN option is also specified. This provides a scalable solution to support a relatively large number of VTNs in an IPv6 network.

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 BCP14 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. New IPv6 Extension Header Option for VTN

A new Hop-by-Hop option type "VTN" is defined to carry the VTN resource identifier and other VTN related information in an IPv6 packet. Its format is shown as below:

<table>
<thead>
<tr>
<th>Option Type</th>
<th>Data Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBCTTTTT</td>
<td>Length</td>
</tr>
<tr>
<td>Option Data</td>
<td>Flags</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>VTN Resource ID</td>
</tr>
</tbody>
</table>

Figure 1. The format of VTN Option

Option Type: 8-bit identifier of the type of option. The type of VTN option is to be assigned by IANA. The bits of the type field are defined as below:

* BB 00 The highest-order 2 bits are set to 00 to indicate that a node which does not recognize this type will skip over it and continue processing the header.

* C 0 The third highest-order bit is set to 0 to indicate this option does not change en route.

* TTTTT To be assigned by IANA.
Opt Data Len: 8-bit unsigned integer indicates the length of the option Data field of this option, in octets. The value of Opt Data Len of VTN option SHOULD be set to 8.

Flags: 8-bit flags field. The most significant bit is defined in this document.

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

* S (Strict Match): The S flag is used to indicate whether the VTN Resource ID MUST be strictly matched for the processing of the packet. When S flag is set to 1, if the VTN resource ID in the VTN option does not match with any of the VTN resource ID provisioned on the outgoing interface of the network node, the packet MUST be dropped. When S flag is set to 0, if the VTN resource ID in the VTN option does not match with any of the VTN resource ID provisioned on the outgoing interface of the network node, the packet SHOULD be forwarded using the default set of network resource on the outgoing interface.

* U (Unused): These flags are reserved for future use. They SHOULD be set to 0 on transmission and MUST be ignored on receipt.

Reserved: 3-octet field reserved for future use. SHOULD be set to 0 on transmission and MUST be ignored on receipt.

VTN Resource ID: 4-octet identifier which uniquely identifies the set of network resources allocated to a VTN.

Note that, if a deployment found it useful, the four-octet VTN Resource ID field may be derived from the four-octet Single Network Slice Selection Assistance Information (S-NSSAI) defined in 3GPP [TS23501].

3. Procedures

As the VTN option needs to be processed by each node along the path for VTN-specific forwarding, it MUST be carried in IPv6 Hop-by-Hop options header.
3.1. Adding VTN Option to Packet

When an ingress node of an IPv6 domain receives a packet, according to the traffic classification or mapping policy, the packet is steered into one of the VTNs in the network, then the packet MUST be encapsulated in an outer IPv6 header, and the Resource ID of the VTN which the packet is mapped to MUST be carried in the VTN option of the Hop-by-Hop Options header associated with the outer IPv6 header.

3.2. VTN based Packet Forwarding

On receipt of a packet with the VTN option, each network node which can process the VTN option in fast path MUST use the VTN Resource ID to determine the set of local network resources allocated to the VTN for packet processing. The packet forwarding behavior is based on both the destination IP address and the VTN Resource ID. More specifically, the destination IP address is used to determine the next-hop and the outgoing interface, and VTN Resource ID is used to determine the set of network resources on the outgoing interface which are allocated to the VTN for processing and sending the packet. If the VTN Resource ID does not match with any of the VTN Resource ID provisioned on the outgoing interface, the S flag in the VTN option is used to determine whether the packet is dropped or forwarded using the default set of network resources of the outgoing interface. The Traffic Class field of the outer IPv6 header MAY be used to provide differentiated treatment for packets which belong to the same VTN. The egress node of the IPv6 domain MUST decapsulate the outer IPv6 header and the Hop-by-Hop Options header which includes the VTN option.

In the forwarding plane, there can be different approaches of partitioning the local network resources and allocating them to different VTNs. For example, on one physical interface, a subset of the forwarding plane resources (e.g. bandwidth and the associated buffer and queuing resources) can be allocated to a particular VTN and represented as a virtual sub-interface with reserved bandwidth resource. In packet forwarding, the IPv6 destination address of the received packet is used to identify the next-hop and the outgoing layer-3 interface, and the VTN Resource ID is used to further identify the virtual sub-interface on the outgoing interface which is associated with the VTN.
Network nodes which do not support the processing of Hop-by-Hop Options header SHOULD ignore the Hop-by-Hop options header and forward the packet only based on the destination IP address. Network nodes which support Hop-by-Hop Options header, but do not support the VTN option SHOULD ignore the VTN option and continue to forward the packet based on the destination IP address. The network node MAY process the rest of the Hop-by-Hop options in the Hop-by-Hop Options header.

4. Operational Considerations

As described in [RFC8200], network nodes may be configured to ignore the Hop-by-Hop Options header, drop packets containing a Hop-by-Hop Options header, or assign packets containing a Hop-by-Hop Options header to a slow processing path. [I-D.ietf-6man-hbh-processing] specifies the modified procedures for the processing of IPv6 Hop-by-Hop Options header. Operator needs to make sure that all the network nodes involved in a VTN can either process the Hop-by-Hop Options header in the fast path, or ignore the Hop-by-Hop Options header. Since a VTN is associated with a logical network topology, it is practical to ensure that all the network nodes involved in that logical topology support the processing of the Hop-by-Hop Options header and the VTN option. In other word, packets steered into a VTN MUST NOT be dropped due to the existence of the Hop-by-Hop Options header. It is RECOMMENDED to configure all the network nodes involved in a VTN to process the Hop-by-Hop Options header and the VTN option if there is a nob for this.

5. IANA Considerations

This document requests IANA to assign a new option type from "Destination Options and Hop-by-Hop Options" registry.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA</td>
<td>VTN Option</td>
<td>this document</td>
</tr>
</tbody>
</table>

6. Security Considerations

The security considerations with IPv6 Hop-by-Hop Options header are described in [RFC8200], [RFC7045], [RFC9098] and [I-D.ietf-6man-hbh-processing]. This document introduces a new IPv6 Hop-by-Hop option which is either processed in the fast path or ignored by network nodes, thus it does not introduce additional security issues.

7. Contributors
8. Acknowledgements

The authors would like to thank Juhua Xu, James Guichard, Joel Halpern, Tom Petch, Aijun Wang, Zhengiang Li, Tom Herbert, Adrian Farrel and Eric Vyncke for their review and valuable comments.

9. References

9.1. Normative References

[I-D.ietf-6man-hbh-processing]

[I-D.ietf-teas-enhanced-vpn]


9.2. Informative References
[I-D.dong-teas-nrp-scalability]
Dong, J., Li, Z., Gong, L., Yang, G., Guichard, J. N.,
Mishra, G., Qin, F., Saad, T., and V. P. Beeram,
"Scalability Considerations for Network Resource
Partition", Work in Progress, Internet-Draft, draft-dong-
teas-nrp-scalability-02, 16 May 2022,
<https://www.ietf.org/archive/id/draft-dong-teas-nrp-
scalability-02.txt>.

[I-D.ietf-lsr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and
A. Guiko, "IGP Flexible Algorithm", Work in Progress,
Internet-Draft, draft-ietf-lsr-flex-algo-20, 18 May 2022,
<https://www.ietf.org/archive/id/draft-ietf-lsr-flex-algo-
20.txt>.

[I-D.ietf-teas-ietf-network-slices]
Farrel, A., Drake, J., Rokui, R., Homma, S., Makhijani,
K., Contreras, L. M., and J. Tantsura, "Framework for IETF
Network Slices", Work in Progress, Internet-Draft, draft-
ietf-teas-ietf-network-slices-12, 30 June 2022,
<https://www.ietf.org/archive/id/draft-ietf-teas-ietf-
network-slices-12.txt>.

Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF",
RFC 4915, DOI 10.17487/RFC4915, June 2007,

Topology (MT) Routing in Intermediate System to
Intermediate Systems (IS-ISs)", RFC 5120,
DOI 10.17487/RFC5120, February 2008,

of IPv6 Extension Headers", RFC 7045,
DOI 10.17487/RFC7045, December 2013,

[RFC9098]  Gont, F., Hilliard, N., Doering, G., Kumari, W., Huston,
G., and W. Liu, "Operational Implications of IPv6 Packets
with Extension Headers", RFC 9098, DOI 10.17487/RFC9098,


Authors’ Addresses

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Road
Beijing
100095
China
Email: jie.dong@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Campus, No. 156 Beiqing Road
Beijing
100095
China
Email: lizhenbin@huawei.com

Chongfeng Xie
China Telecom
China Telecom Beijing Information Science & Technology, Beiqijia
Beijing
102209
China
Email: xiechf@chinatelecom.cn

Chenhao Ma
China Telecom
China Telecom Beijing Information Science & Technology, Beiqijia
Beijing
102209
China
Email: machh@chinatelecom.cn
IPv6 Hop-by-Hop Options Processing Procedures
draft-ietf-6man-hbh-processing-01

Abstract

This document specifies procedures for how IPv6 Hop-by-Hop options are processed. It modifies the procedures specified in the IPv6 Protocol Specification (RFC8200) to make processing of IPv6 Hop-by-Hop options practical with the goal of making IPv6 Hop-by-Hop options useful to deploy and use in the Internet. When published, this document updates RFC8200.

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 8 January 2023.

Copyright Notice

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

This document specifies procedures for how IPv6 Hop-by-Hop options are processed. It modifies the procedures specified in the IPv6 Protocol Specification (RFC8200) to make processing of IPv6 Hop-by-Hop options practical with the goal of making IPv6 Hop-by-Hop options useful to deploy and use in the Internet.

The editors focus for this document is to set a lower bound expectation for the minimum number of hop-by-hop options that a node supports. This document does not discuss an upper bound. This topic is discussed in [I-D.ietf-6man-eh-limits].

When published this document updates [RFC8200].

The current list of defined Hop-by-Hop options can be found at [IANA-HBH].
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. Terminology

This document uses the following loosely defined terms:

* **Forwarding Plane:** IPv6 hosts exchange user data through the forwarding plane. User data is processed by its recipient (i.e., an IPv6 host). User data can traverse intermediate nodes (i.e., routers) between its source and its destination. These intermediate nodes process metadata contained in packet headers. However, they do not process information contained in packet payloads.

* **Control Plane:** IPv6 routers exchange management and routing information with controllers. They also exchange routing information with one another. Management and routing information is processed by its recipient (i.e., an IPv6 router or controller). Management and control information can traverse intermediate nodes (i.e., routers) between its source and its destination. These intermediate nodes process metadata contained in packet headers. However, they do not process information contained in packet payloads. So, from their perspective, this information is user data.

* **Fast Path:** A path through a router that is optimized for forwarding packets without processing their payloads. The Fast Path may be supported by Application Specific Integrated Circuits (ASICs), Network Processor (NP), or other special purpose hardware. This is the usual processing path within a router taken by the forwarding plane.

* **Slow Path:** A path through a router that is capable of general purpose processing and is not optimized for any particular function. This is processing path is used for packets that require special processing or differ from assumptions made in Fast Path heuristics, or to process router control protocols used by the control plane.
NOTE: This distinct separation between hardware and software processing from [RFC6398] does not apply to all router architectures. However, a router that performs all or most processing in software might still incur more processing cost when providing special processing (aka Slow Path).

[RFC6192] is an example of how designs can separate control plane (Slow Path) and forwarding plane (Fast Path) functions.

4. Background

In the first version of the IPv6 specification, Hop-by-Hop options were required to be processed by all nodes: routers and hosts. This proved to not be practical in high speed routers due to several factors, including:

* Inability to process the hop-by-hop options at full the forwarding rate (e.g., routers with no support on the Fast Path).

* Hop-by-Hop options would be sent to the Slow Path. This could degrade the a router’s performance and it’s ability to process important control traffic.

* A mechanism that forces packets from any source to the routers "Slow Path" could be exploited as a Denial of Service attack against the router.

* Packets could contain multiple Hop-by-Hop options making the previous issues worse by increasing the complexity required to process them.

When the IPv6 Specification was updated and published in July 2017 as [RFC8200], the procedures relating to hop-by-hop options were as follows:

Extension headers (except for the Hop-by-Hop Options header) are not processed, inserted, or deleted by any node along a packet’s delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header.
The Hop-by-Hop Options header is not inserted or deleted, but may be examined or processed by any node along a packet’s delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header. The Hop-by-Hop Options header, when present, must immediately follow the IPv6 header. Its presence is indicated by the value zero in the Next Header field of the IPv6 header.

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

The changes meant that an implementation complied with the IPv6 specification even if it did not process hop-by-hop options, and that it was expected that routers would add configuration information to control which hop-by-hop options they would process.

The text regarding processing Hop-by Hop Options in [RFC8200] was not intended to change the processing of Hop-by-Hop options. It only documented how they were being used in the Internet at the time RFC8200 was published. This was a constraint on publishing the IPv6 specification as an IETF Standard.

The main issues remain:

* Routers are commonly configured to drop transit packets containing hop-by-hop options that would have be processed in the Slow Path. This behavior is seen as protecting against a denial of service attack on the router. A survey in 2015 reported a high loss rate in transit ASs for packets with HBH options [RFC7872]. The operational implications of IPv6 Packets that set extension headers is discussed in [RFC9098].

* Allowing multiple hop-by-hop options in a single packet makes it even more expensive in router resources to process these packets. It adds complexity to the number of permutations that might need to be processed.

* Any mechanism that can be used to force packets into the router’s Slow Path can be exploited as a denial of service attack on a transit router by saturating the resources needed for router management protocols (e.g., routing protocols, network management protocols, etc.) that may cause the router to fail. This issue for the Router Alert option, which intentionally places packets on the Slow Path, is discussed in [RFC6398]. Section 3 of that RFC includes a good summary:
"In a nutshell, the IP Router Alert Option does not provide a convenient universal mechanism to accurately and reliably distinguish between IP Router Alert packets of interest and unwanted IP Router Alert packets. This, in turn, creates a security concern when the IP Router Alert Option is used, because, short of appropriate router-implementation-specific mechanisms, the router Slow Path is at risk of being flooded by unwanted traffic."

There has been research that discussed the general problem with dropping packets containing IPv6 extension headers, including the Hop-by-Hop Options header. For example [Hendriks] states that "dropping all packets with Extension Headers, is a bad practice", and that "The share of traffic containing more than one EH however, is very small. For the design of hardware able to handle the dynamic nature of Extension Headers we therefore recommend to support at least one EH".

The authors expectations are that some hop-by-hop options will be processed across the Internet while others will only be processed in a limited domain (e.g., where there is a specific service made available in that network segment that relies on one or more hop-by-hop options).

This document defines a set of procedures for the hop-by-hop option header that make the processing of hop-by-hop options practical in modern transit routers.

5. Hop-by-Hop Header Processing Procedures

This section describes several changes to [RFC8200].

5.1. Hop-by-Hop Options Per Packet

The Hop-by-Hop Option Header as defined in Section 4.3 of [RFC8200] is identified by a Next Header value of 0 in the IPv6 header. Section 4.1 of [RFC8200] requires a Hop-by-Hop Options header to appear immediately after the IPv6 header. [RFC8200] also requires that a Hop-by-Hop Options header can only appear once in a packet.

The Hop-by-Hop Options Header as defined in [RFC8200] can contain one or more Hop-by-Hop options. This document updates [RFC8200] that a node MUST process the first Option in the Hop-by-Hop Header at full forwarding rate the (e.g. on the router’s Fast Path) and MAY process additional Hop-by-Hop Options if configured to do so. The motivation for this change is to simplify the processing of Hop-by-Hop options as a part of normal forwarding.
Nodes creating packets with a Hop-by-Hop option headers SHOULD include a single Hop-by-Hop Option in the packet and MAY include more based on local configuration.

If there are more than one Hop-by-Hop options in the Hop-by-Hop Options header, the node MAY skip the rest of the options without having to examine these options using the "Hdr Ext Len" field in the Hop-by-Hop Options header. This field specifies the length of the Option Header in 8-octet units. The additional options do not need to be processed or verified.

5.2. Hop-by-Hop Headers Processing

Nodes that implement a differentiation between a Fast Path and a Slow Path MUST process all (with one exception noted below) Hop-by-Hop options in the Fast Path. The one exception to this is the Router Alert Option [RFC2711]. See Section 5.3 for discussion of the Router Alert.

If the node can not process an option at the full forwarding rate, it MUST behave as if it does not recognize the Option Type (as described in the next paragraph).

Section 4.2 of [RFC8200] defines the Option Type identifiers as internally encoded such that their highest-order 2 bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type. The text is:

00 - skip over this option and continue processing the header.

01 - discard the packet.

10 - discard the packet and, regardless of whether or not the packet’s Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet’s Source Address, pointing to the unrecognized Option Type.

11 - discard the packet and, only if the packet’s Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet’s Source Address, pointing to the unrecognized Option Type.

This document modifies this behaviour for the "10" and "11" values that the node MAY send an ICMP Parameter Problem, Code 2, message to the packet’s Source Address, pointing to the unrecognized Option Type. The modified text for "10" and 11" values is:
10 - discard the packet and, regardless of whether or not the packet’s Destination Address was a multicast address, MAY send an ICMP Parameter Problem, Code 2, message to the packet’s Source Address, pointing to the unrecognized Option Type.

11 - discard the packet and, only if the packet’s Destination Address was not a multicast address, MAY send an ICMP Parameter Problem, Code 2, message to the packet’s Source Address, pointing to the unrecognized Option Type.

The motivation for this change is to loosen the requirement to send ICMPv6 Parameter Problem messages by simplifying what the router needs to do when it performs forwarding of an Option Type it does not recognize.

When an ICMP Parameter Problem, Code 2, message is delivered to the source, the source can become aware that at least one node on the path has failed to recognize the option.

5.3. Router Alert Option

The Router Alert option [RFC2711] purpose is to tell the node that the packet needs additional processing on the Slow Path.

The Router Alert option includes a two octet Value field that describes the protocol that is carried in the packet. The current values can be found in the IANA Router Alert Value registry [IANA-RA].

DISCUSSION

The Router Alert Option is a problem since it’s function is to do what this specification is proposing to eliminate, that is, process the packet in the Slow Path. One approach would be to deprecate it as it’s usage appears to be limited and packets containing Hop-by-Hop options are frequently dropped. Deprecation would allow current implementations to continue and it’s use could be phased out over time.

The authors current thinking is that the Router Alert function may have reasonable potential use for new functions that have to be processed in the Slow Path. We think that keeping it as the single exception for Slow Path processing with the following restrictions is a reasonable compromise to allow future flexibility. These are compatible with Section 5 of [RFC6398].
A Fast Path implementation SHOULD verify that a Router Alert contains a protocol, as indicated by the Value field in the Router Alert option, that is configured as a protocol of interest to that router. A verified packet SHOULD be sent on the Slow Path for processing [RFC6398]. Otherwise, the router implementation SHOULD forward within the Fast Path (subject to all normal policies and forwarding rules). As specified in [RFC2711] the top two bits of Option Type for the Router Alert option are always set to "00" indicating the node should skip over this option and continue processing the header in this case.

Implementations of the IP Router Alert Option SHOULD offer the configuration option to simply ignore the presence of "IP Router Alert" in IPv4 and IPv6 packets" [RFC6398].

A node that is configured to process a Router Alert option using the Slow Path MUST protect itself from infrastructure attack that could result from processing on the Slow Path. This might include some combination of access control list to only permit from trusted nodes, rate limiting of processing, or other methods [RFC6398].

5.4. Configuration

Section 4 of [RFC8200] allows for a router to control it’s processing of IPv6 Hop-by-Hop options by local configuration. The text is:

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

A possible approach to implementing this is to maintain a lookup table based on Option Type of the IPv6 options that are supported in the Fast Path. This would allow for a node to quickly determine if an option is supported and can be processed. If the option is not supported, then the node processes it as described in Section 5.2 of this document.

A node configured not to process HBH options, MUST drop the packet if the top two bits of the Option Type field of the first HBH option is non-zero.

The actions of the lookup table SHOULD be configurable by the operator of the router.
6. New Hop-by-Hop Options

Any new IPv6 Hop-by-Hop option designed in the future should be designed to be processed at full forwarding rate (e.g., on a router’s Fast Path). New options SHOULD NOT be defined that are not expected to be executed at full forwarding rates. New Hop-by-Hop options SHOULD have the following characteristics:

* Straight forward to process. That is, they should be designed to keep the time to process low.

* New Hop-by-Hop options should be designed to be the first option in a Hop-by-Hop options header.

* The size of an option should not extend beyond what can be reasonably expected to be executed at full forwarding rate (e.g., forwarded on a router’s fast path).

Any new Hop-by-Hop option that is standardized that does not meet these criteria needs to explain in detail in its specification why this can not be accomplished and that there is a reasonable expectation that it can be proceed at full forwarding rate.

7. IANA Considerations

There are no actions required for IANA defined in this document.

8. Security Considerations

Security issues with IPv6 Hop-by-Hop options are well known and have been documented in several places, including [RFC6398], [RFC6192], and [RFC9098]. The main issue, as noted in Section 4, is that any mechanism that can be used to force packets into the router’s Slow Path can be exploited as a denial of service attack on a transit router by saturating the resources need for router management protocols (e.g., routing protocols, network management protocols, etc.) that may cause the router to fail. Due to this it’s common for transit routers to drop packets with Hop-by-Hop options headers.

While Hop-by-Hop options are not required to be processed in the Slow Path, the Router Alert options is designed to do just that.

This document changes the way Hop-by-Hop options are processed in several ways that significantly reduces the attack surface. These changes include:
All Hop-by-Hop options (with one exception) must be processed in the Fast Path. Only one HBH Option MUST be processed and additional HBH Options MAY be processed based on local configuration.

Only the Router Alert option can be processed in the Slow Path, and the router must be configured to do so.

Added criteria to allow control over how Router Alert options are processed and that a node configured to support these options must protect itself from attacks using the Router Alert.

Limited the default number of Hop-by-Hop options that can be in a packet to a single Hop-by-Hop option.

Additional Hop-by-Hop options MAY be included, based on local configuration. Although nodes only process these additional Hop-by-Hop Options if configured to do so.

Added restrictions to any future new Hop-by-Hop options that limit their size and computational requirements.

The authors believe that these changes significantly reduces the security issues relating to IPv6 Hop-by-Hop options and will enable them to be used safely in the Internet.

9. Acknowledgments

Helpful comments were received from Brian Carpenter, Ron Bonica, Ole Troan, Mark Heard, Tom Herbert, Cheng Li, Eric Vyncke, Greg Mirksy, Xiao Min, Fernando Gont, Darren Dukes, [your name here], and other members of the 6MAN working group.

10. Change log [RFC Editor: Please remove]

draft-ietf-6man-hbh-processing-01, 2022-June-15:

* Fixed typo in last paragraph of Section 5.2
* Revised text in Section 4 to reflect constraints on publishing RFC8200.
* Changed text in Section 6 that new options SHOULD NOT (from MUST NOT) be defined that require that are not expected to be excepted at full forwarding rates.
* Added reference to RFC7872 in Section 4.
* Added text to Section 1 that the focus of this document is to set a mimium bound on the number of Hop-by-Hop Options a node should process.
* Added text to Section 4 that the authors some Hop-by-Hop options will be supported Internet wide, and others only in limited domains.
* Editorial changes.

draft-ietf-6man-hbh-processing-00, 2022-January-29:
* 6MAN Working Group Draft
* Reworked text to talk about processing HBH options at full forwarding rates, instead of "fast path"
* Revised Section 6 "New Hop-by-Hop Options" to allow variable sized HBH options, remove specific length requirements, and other clarifications.
* Editorial changes.

draft-hinden-6man-hbh-processing-01, 2021-June-2:
* Expanded terminology section to include Forwarding Plane and Control Plane.
* Changed draft that only one HBH Option MUST be processed and additional HBH Options MAY be processed based on local configuration.
* Clarified that all HBH options (with one exception) must be processed on the Fast Path.
* Kept the Router Alert options as the single exception for Slow Path processing.
* Rewrote and expanded section on New Hop-by-Hop Options.
* Removed requirement for HBH Option size and alignment.
* Removed sections evaluating currently defined HBH Options.
* Added content to the Security Considerations section.
* Added people to the acknowledgements section.
* Numerous editorial changes

draft-hinden-6man-hbh-processing-00, 2020-Nov-29:
* Initial draft.

11. Normative References

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
12. Informative References


Authors’ Addresses

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
United States of America
Email: bob.hinden@gmail.com

Godred Fairhurst
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen
AB24 3UE
United Kingdom
Email: gorry@erg.abdn.ac.uk
Abstract

The data plane for Segment Routing over IPv6 (SRv6) [RFC8754] is built using IPv6 as the underlying forwarding plane. Due to this underlying use of IPv6, Segment Identifiers (SIDs) used by SRv6 can resemble IPv6 addresses and behave like them [RFC8754][RFC8986] while exhibiting slightly different behaviors in some situations. This document intends to explore the characteristics of SRv6 SIDs and to clarify the relationship of SRv6 SIDs to the IPv6 Addressing Architecture [RFC4291].
1. **Introduction**

Segment Routing over IPv6 (SRv6) [RFC8754] uses IPv6 as the underlying data plane. In SRv6, SR source nodes initiate packets with a segment identifier in the Destination Address of the IPv6 header, and SR segment endpoint nodes that process a local segment present the Destination Address of an IPv6 header. Thus Segment Identifiers (SIDs) in SRv6 can and do appear in the Destination Address field of IPv6 datagrams by design.

2. **Terminology**

The following terms are used as defined in [RFC8402].

* Segment Routing (SR)

* SR Domain

* Segment

* Segment Identifier (SID)
3. SRv6 SIDs and the IPv6 addressing architecture

[RFC8754] defines the Segment List of the SRH as a contiguous array of 128-bit IPv6 addresses, and that each of the elements in this list are SIDs. But all of these elements are not necessarily made equal. Some of these elements may represent a local interface as described in Section 4.3 of [RFC8754] as "A FIB entry that represents a local interface, not locally instantiated as an SRv6 SID". From this it follows that all the SIDs that appear in the SRH are not SRv6 SIDs as defined by [RFC8402].

It is also fairly clear that the non-SRv6-SID elements that appear in the SRH SID list are simply IPv6 addresses assigned to local interfaces and MUST conform to [RFC4291]. So, the following discussions are intended to be applicable solely to SRv6 SIDs that are not assigned to local interfaces.
One of the key questions to address is how these SRv6 SIDs appearing as IPv6 Destination Addresses are perceived and treated by "transit nodes" (that are not required to be capable of processing a Segment or the Segment Routing Header).

Section 3.1. of [RFC8986] describes the format of an SRv6 SID as composed of three parts LOC:FUNCT:ARG, where a locator (LOC) is encoded in the L most significant bits of the SID, followed by F bits of function (FUNCT) and A bits of arguments (ARG). Such a SID is assigned to a node within a prefix defined as a Locator of length L. When an SRv6 SID occurs in the IPv6 destination address field of an IPv6 header, only the longest match prefix corresponding to the locator is used to forward the packet to the node identified by the Locator.

It is clear that this format for SRv6 SIDs is not compliant with the requirements set forth in [RFC4291] for IPv6 addresses but it is also clear that SRv6 SIDs are not intended for assignment onto interfaces on end hosts. They look and act similar to other mechanisms that use IPv6 addresses with different formats such as [RFC6052] that defines the IPv6 Addressing of IPv4/IPv6 Translators and [RFC7343] that describes ORCHIDv2 (a cryptographic hash identifier format).

While looking at the transit nodes it becomes apparent that these addresses are used purely for routing and not for packet delivery to end hosts. Hence the relevant standard to apply here is [RFC7608] that allows the use of variable length prefixes in forwarding while explicitly decoupling IPv6 routing and forwarding from the IPv6 address/prefix semantics described in [RFC4291]. Please note that [RFC7608] does not override the rules in [RFC4291], but merely limits where their impact is observed.

Furthermore, in the SRv6 specifications, all SIDs assigned within a given Locator prefix are located inside the node identified by Locator. Therefore there does not appear to be a conflict with section 2.6.1 of [RFC4291] since subnet-router anycast addresses are neither required nor useful within a node.

4. Special Considerations for Compressed SIDs

The C-SID document [I-D.filsfilscheng-spring-srv6-srh-compression] describes how to use a single entry in the SRH list as a container for multiple SIDs and defines a few flavors of how to do so. A node taking part in this mechanism accomplishes this by using the ARG part [RFC8986] of the Destination address field of the IPv6 header to come up with a new Destination address in some of these flavors. i.e. The destination address field of the packet changes at a segment endpoint in a way similar to how the address changes as the result of
processing a segment in the SRH.

One key thing to note in here is that the Locator Block at the beginning of the address does not get modified by the operations needed for supporting compressed SIDs. As we have established that the SRv6 SIDs are being treated simply as routing prefixes on transit nodes this does not constitute a modification to the IPv6 data plane on such transit nodes and any changes are restricted to SR aware nodes.

4.1. Open Issues to be Addressed with C-SIDs

There are a few issues that need to be addressed in the C-SID draft prior to its publication as RFC:

* This draft needs to provide an updated definition for the SegmentsLeft field of the SRH since the current definition in [RFC8754][RFC8200] no longer holds true in the presence of C-SIDs.

* In some cases it is possible that the SR policy can be expressed purely with C-SIDs without requiring an SRH. In this case, to allow the SR domain to fail closed, some form of filtering based on the LOC part of the SRv6 SID is required as relying purely on the presence of an SRH will not be sufficient.

* The use of C-SIDs might cause some difficulty in troubleshooting error conditions signaled by ICMPv6. Section 5.4 of [RFC8754] describes the ICMPv6 error processing that is required to be performed on the SR Source Nodes to correlate packets since the Destination Address field of the packet changes in flight. Similar logic needs to be specified for SR Source Nodes that use C-SIDs to determine the destination address for use by protocol-error handlers.

4.2. Applicability to other forms of compressed SIDs

The spring working group is in the process of analyzing multiple mechanisms for compressing the SRv6 SID list as described in [I-D.ietf-spring-compression-analysis]. Even though this document focuses on [I-D.filsfilscheng-spring-srv6-srh-compression], the considerations specified in this document might also be applicable to the other mechanisms being analyzed and compared.
5. Allocation of a Global Unicast Prefix for SIDs

All of the SRv6 related specifications discussed above are intended to be applicable to a contained SR Domain or between collaborating SR Domains. Hence the behavior of SRv6 SIDs is visible purely within the SR domain and they would be treated solely as IPv6 routing prefixes by nodes that are not SR aware.

As an added factor of safety, it might be prudent to allocate some address space that explicitly signals that the addresses within that space are not intended to comply with [RFC4291]. As described in Section 3 above, there is precedent for mechanisms that use IPv6 addresses in a manner different from that specified in [RFC4291]. This would be useful in identifying and potentially filtering packets at the edges of the SR Domains as described in Section 4.1.

The SRv6 operational community, which is the first intended user of this block, is requested to come up with conventions and guidelines for the use of this newly allocated address block in line with their requirements.

6. IANA Considerations

IANA is requested to assign a /16 address block for the purposes described in Section 5 out of the "Reserved by IETF" range defined in the Internet Protocol Version 6 Address Space registry.

7. Security Considerations

The security considerations for the use of Segment Routing [RFC8402], SRv6 [RFC8754], and SRv6 network programming [RFC8986] apply to the use of these addresses. The use of IPv6 tunneling mechanisms (including SRv6) also brings up additional concerns such as those described in [RFC6169].

8. Acknowledgments

The author would like to extend a special note of thanks to Brian Carpenter and Erik Kline for their precisely summarized thoughts on this topic that provided the seed of this draft. The author would also like to thank Andrew Alston, Ron Bonica, Bruno Decraene, Darren Dukes, Clarence Filsfils, Jim Guichard, Joel Halpern, Bob Hinden, Alvaro Retana, Ole Troan, Eric Vyncke and Jen Linkova for their ideas and comments to improve this document.

9. References

9.1. Normative References
9.2. Informative References
[I-D.filsfilscheng-spring-srv6-srh-compression]
Cheng, W., Filsfils, C., Li, Z., Decraene, B., Cai, D.,
Voyer, D., Clad, P., Zadok, S., Guichard, J., Liu, A.,
Raszuk, R., and C. Li, "Compressed SRv6 Segment List
Encoding in SRH", Work in Progress, Internet-Draft, draft-
filsfilscheng-spring-srv6-srh-compression-02, 28 July
2021, <https://www.ietf.org/archive/id/draft-
filsfilscheng-spring-srv6-srh-compression-02.txt>.

[I-D.ietf-spring-compression-analysis]
Bonica, R., Cheng, W., Dukes, D., Henderickx, W., Li, C.,
Peng, S., and C. Xie, "Compressed SRv6 SID List Analysis",
Work in Progress, Internet-Draft, draft-ietf-spring-
compression-analysis-00, 27 September 2021,
<https://www.ietf.org/archive/id/draft-ietf-spring-
compression-analysis-00.txt>.

[RFC6052]  Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X.
Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052,
DOI 10.17487/RFC6052, October 2010,

Concerns with IP Tunneling", RFC 6169,
DOI 10.17487/RFC6169, April 2011,

[RFC7343]  Laganier, J. and F. Dupont, "An IPv6 Prefix for Overlay
Routable Cryptographic Hash Identifiers Version 2
(ORCHIDv2)", RFC 7343, DOI 10.17487/RFC7343, September

Author's Address
Suresh Krishnan
Cisco
Email: suresh.krishnan@gmail.com
Improving the Robustness of Stateless Address Autoconfiguration (SLAAC) to Flash Renumbering Events
draft-ietf-6man-slaac-renum-03

Abstract

In renumbering scenarios where an IPv6 prefix suddenly becomes invalid, hosts on the local network will continue using stale prefixes for an unacceptably long period of time, thus resulting in connectivity problems. This document improves the reaction of IPv6 Stateless Address Autoconfiguration to such renumbering scenarios.

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 8 December 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) 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.
Table of Contents

1. Introduction .................................................. 3
2. Terminology .................................................... 3
3. SLAAC reaction to Flash-renumbering Events .................. 4
   3.1. Renumbering without Explicit Signaling ................. 4
   3.2. Renumbering with Explicit Signaling .......... 5
4. Improvements to Stateless Address Autoconfiguration (SLAAC) ........................................... 6
   4.1. More Appropriate Lifetime Values ................. 7
   4.1.1. Router Configuration Variables ................. 7
   4.2. Honor Small PIO Valid Lifetimes ................. 8
   4.3. Interface Initialization ......................... 9
   4.4. Conveying Information in Router Advertisement (RA) Messages ............................................. 10
   4.5. Recovery from Stale Configuration Information without Explicit Signaling ................. 10
5. IANA Considerations ........................................... 10
6. Implementation Status ......................................... 10
   6.1. More Appropriate Lifetime Values ................. 10
   6.1.1. Router Configuration Variables ................. 10
   6.2. Honor Small PIO Valid Lifetimes ................. 11
   6.2.1. Linux Kernel ........................................... 11
   6.2.2. NetworkManager ........................................... 11
   6.3. Conveying Information in Router Advertisement (RA) Messages ............................................. 11
   6.4. Recovery from Stale Configuration Information without Explicit Signaling ................. 11
   6.4.1. dhcpcd(8) ........................................... 11
6.5. Other mitigations implemented in products ................. 11
7. Security Considerations ........................................ 12
8. Acknowledgments ................................................ 12
9. References ...................................................... 13
   9.1. Normative References ...................................... 13
   9.2. Informative References ...................................... 14
Appendix A. Analysis of Some Suggested Workarounds ............. 15
   A.1. On a Possible Reaction to ICMPv6 Error Messages ......... 16
   A.2. On a Possible Improvement to Source Address Selection ...... 16
Authors’ Addresses ................................................ 18
1. Introduction

IPv6 network renumbering is expected to take place in a planned manner, with old/stale prefixes being phased-out via reduced prefix lifetimes while new prefixes (with normal lifetimes) are introduced. However, there are a number of scenarios that may lead to the so-called "flash-renumbering" events, where the prefix being employed on a network suddenly becomes invalid and replaced by a new prefix [RFC8978]. In such scenarios, hosts on the local network will continue using stale prefixes for an unacceptably long period of time, thus resulting in connectivity problems. [RFC8978] discusses this problem in detail.

In some scenarios, the local router producing the network renumbering event may try to deprecate the currently-employed prefixes (thus explicitly signaling the network about the renumbering event), whereas in other scenarios it may be unaware about the renumbering event and thus unable signal hosts about it.

From the perspective of a Stateless Address Autoconfiguration (SLAAC) host, there are two different (but related) problems to be solved:

* Avoiding the use of stale addresses for new communication instances

* Performing "garbage collection" for the stale prefixes (and related network configuration information)

Clearly, if a host has both working and stale addresses, it is paramount that it employs working addresses for new communication instances. Additionally, a host should also perform garbage collection for the stale prefixes/addresses, since they not only tie system resources, but also prevent communication with the new "owners" of the stale prefixes.

2. Terminology

The term "globally reachable" is used in this document as defined in [RFC8190].

The term "Global Unicast Address" (or its acronym "GUA") is used throughout this document to refer to "globally reachable" [RFC8190] addresses. That is, when used throughout this document, GUAs do NOT include Unique Local Addresses (ULAs) [RFC4193]. Similarly, the term "Global Unicast prefix" (or "GUA prefix") is employed throughout this document to refer to network prefixes that specify GUAs, and does NOT include the ULA prefix (FC00::/7) [RFC4193].
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. SLAAC reaction to Flash-renumbering Events

As noted in Section 1, in some scenarios the router triggering the renumbering event may be able to explicitly signal the network about this event, while in other scenarios the renumbered hosts may need to infer a renumbering event is taking place. The following subsections analyze specific considerations for each of these scenarios.

3.1. Renumbering without Explicit Signaling

In the absence of explicit signalling from SLAAC routers (such as sending Prefix Information Options (PIOs) with small lifetimes to deprecate the stale prefixes), stale prefixes will remain preferred and valid according to the Preferred Lifetime and Valid Lifetime values (respectively) of the last received PIO. IPv6 SLAAC employs the following default values for PIOs:

* Preferred Lifetime (AdvPreferredLifetime): 604800 seconds (7 days)
* Valid Lifetime (AdvValidLifetime): 2592000 seconds (30 days)

This means that, in the absence of explicit signaling by a SLAAC router to deprecate a prefix, it will take a host 7 days (one week) to deprecate the corresponding addresses, and 30 days (one month) to eventually remove any addresses configured for the stale prefix. Clearly, for any practical purposes, employing such long default values is the equivalent of not using any timers at all, since taking 7 days or 30 days (respectively) to recover from a network problem is simply unacceptable.

Use of more appropriate timers in Router Advertisement messages can help limit the amount of time that hosts will maintain stale configuration information. Additionally, hosts are normally in a position to infer that a prefix has become stale -- for example, if a given router ceases to advertise an existing prefix and at the same time starts to advertise a new prefix.

Section 4.1.1 recommends the use of more appropriate default lifetimes for PIOs, while Section 4.5 specifies a local policy that SLAAC hosts can implement to heuristically infer that network configuration information has changed, such that stale configuration information can be phased out.
3.2. Renumbering with Explicit Signaling

In scenarios where a local router is aware about the renumbering event, it may try to phase out the stale network configuration information. In these scenarios, there are two aspects to be considered:

* The amount of time during which the router should continue trying to deprecate the stale network configuration information

* The ability of SLAAC hosts to phase out stale configuration in a timelier manner.

In the absence of Router Advertisements (RAs) that include PIOs that would reduce the Valid Lifetime and Preferred Lifetime of a prefix, hosts would normally employ the lifetime values from PIO options of the last received RA messages. Since the network could be partitioned for an arbitrarily long period of time, a router would need to try to deprecate a prefix for the amount of time employed for the "Preferred Lifetime", and try to invalidate the prefix for the amount of time employed for the "Valid Lifetime" (see Section 12 of [RFC4861]).

NOTE:

Once the number of seconds in the original "Preferred Lifetime" have elapsed, all hosts would have deprecated the corresponding addresses anyway, while once the number of seconds in the "Valid Lifetime" have elapsed, the corresponding addresses would be invalidated and removed.

Thus, use of more appropriate default lifetimes for PIOs, as proposed in Section 4.1.1, would reduce the amount of time a stale prefix would need to be announced as such by a router in order to make sure that it is deprecated/invalidated.

In scenarios where a router has positive knowledge that a prefix has become invalid and thus could signal this condition to local hosts, the current specifications will prevent SLAAC hosts from fully recovering from such stale information. Item "e)" of Section 5.5.3 of [RFC4862] specifies that an RA may never reduce the "RemainingLifetime" to less than two hours. Additionally, if the RemainingLifetime of an address is smaller than 2 hours, then a Valid Lifetime smaller than 2 hours will be ignored. The inability to invalidate a stale prefix would prevent communication with the new "owners" of the stale prefix, and thus is highly undesirable. On the other hand, the Preferred Lifetime of an address *can* be reduced to any value to avoid the use of a stale prefix for new communications.
Section 4.2 updates [RFC4862] such that this restriction is removed, and hosts react to the advertised "Valid Lifetime" (even if it is smaller than 2 hours).

Finally, Section 4.3 recommends that routers disseminate network configuration information when a network interface is initialized, such that possibly new configuration information propagates in a timelier manner.

4. Improvements to Stateless Address Autoconfiguration (SLAAC)

The following subsections update [RFC4861] and [RFC4862], such that the problem discussed in this document is mitigated. The aforementioned updates are mostly orthogonal, and mitigate different aspects of SLAAC that prevent a timely reaction to flash renumbering events.

* Reduce the default Valid Lifetime and Preferred Lifetime of PIOs (Section 4.1.1):

This helps limit the amount of time a host will employ stale information, and also limits the amount of time a router needs to try to obsolete stale information.

* Honor PIOs with small Valid Lifetimes (Section 4.2):

This allows routers to invalidate stale prefixes, since otherwise [RFC4861] prevents hosts from honoring PIOs with a Valid Lifetime smaller than two hours.

* Recommend routers to retransmit configuration information upon interface initialization/reinitialization (Section 4.3):

This helps spread the new information in a timelier manner, and also deprecate stale information via host-side heuristics (see Section 4.5).

* Recommend routers to always send all options (i.e. the complete configuration information) in RA messages, and in the smallest possible number of packets (Section 4.4):

This helps propagate the same information to all hosts, and also allows hosts to better infer that information missing in RA messages has become stale (see Section 4.5).

* Infer stale network configuration information from received RAs (Section 4.5):

Gont, et al. Expires 8 December 2022
This allows hosts to deprecate stale network configuration information, even in the absence of explicit signaling.

4.1. More Appropriate Lifetime Values

4.1.1. Router Configuration Variables

The default values of the Preferred Lifetime and the Valid Lifetime of PIOs are updated as follows:

\[
\text{AdvPreferredLifetime}: \ max(\text{AdvDefaultLifetime}, 3 \times \text{MaxRtrAdvInterval})
\]

\[
\text{AdvValidLifetime}: 2 \times \text{AdvPreferredLifetime}
\]

where:

\text{AdvPreferredLifetime}:
\begin{itemize}
  \item Value to be placed in the "Preferred Lifetime" field of the PIO.
\end{itemize}

\text{AdvValidLifetime}:
\begin{itemize}
  \item Value to be placed in the "Valid Lifetime" field of the PIO.
\end{itemize}

\text{AdvDefaultLifetime}:
\begin{itemize}
  \item Value to be placed in the "Router Lifetime" field of the Router Advertisement message that will carry the PIO.
\end{itemize}

\text{max}():
\begin{itemize}
  \item A function that computes the maximum of its arguments.
\end{itemize}

NOTE:
\[ \text{RFC4861} \] specifies the default value of MaxRtrAdvInterval as 600 seconds, and the default value of AdvDefaultLifetime as \( 3 \times \text{MaxRtrAdvInterval} \). Therefore, when employing default values for MaxRtrAdvInterval and AdvDefaultLifetime, the default values of AdvPreferredLifetime and AdvValidLifetime become 1800 seconds (30 minutes) and 3600 seconds (1 one hour), respectively. We note that when implementing BCP202 \[ \text{RFC7772} \], AdvDefaultLifetime will typically be in the range of 45-90 minutes, and therefore the default value of AdvPreferredLifetime will be in the range 45-90 minutes, while the default value of AdvValidLifetime will be in the range of 90-180 minutes.

RATIONALE:
\begin{itemize}
  \item The default values of the PIO lifetimes should be such that, under normal circumstances (including some packet loss), the associated timers are refreshed/reset, but in the presence of network failures (such as network configuration information
becoming stale), some fault recovering action (such as deprecating the corresponding addresses and subsequently removing them) is triggered.

* In the context of [RFC8028], where it is clear that the use of addresses configured for a given prefix is tied to the next-hop router that advertised the prefix, the "Preferred Lifetime" of a PIO should not be larger than the "Router Lifetime" of Router Advertisement messages. Some leeway should be provided for the "Valid Lifetime" of PIOs, to cope with transient network problems. As a result, this document updates [RFC4861] such that the default Valid Lifetime (AdvValidLifetime) and the default Preferred Lifetime (AdvPreferredLifetime) of PIOs are specified as a function of the "Router Lifetime" (AdvDefaultLifetime) of Router Advertisement messages. In the absence of RAs that refresh information, addresses configured for previously-advertised prefixes become deprecated in a timelier manner, and thus Rule 3 of [RFC6724] will cause other configured addresses (if available) to be preferred.

* The expression above computes the maximum between AdvDefaultLifetime and "3 * MaxRtrAdvInterval" (the default value of AdvDefaultLifetime, as per [RFC4861]) to cope with the case where an operator might simply want to disable one local router for maintenance, without disabling the use of the corresponding prefixes on the local network (e.g., on a multi-router network). [RFC4862] implementations would otherwise deprecate the corresponding prefixes. Similarly, [RFC8028] implementations would likely behave in the same way.

4.2. Honor Small PIO Valid Lifetimes

The entire item "e)" (pp. 19-20) from Section 5.5.3 of [RFC4862] is replaced with the following text:

   e) If the advertised prefix is equal to the prefix of an address configured by stateless autoconfiguration in the list, the valid lifetime and the preferred lifetime of the address should be updated by processing the Valid Lifetime and the Preferred Lifetime (respectively) in the received advertisement.

RATIONALE:

* This change allows hosts to react to the information provided by a router that has positive knowledge that a prefix has become invalid.
* The behavior described in RFC4862 had been incorporated during the revision of the original IPv6 Stateless Address Autoconfiguration specification ([RFC1971]). At the time, the IPNG working group decided to mitigate the attack vector represented by Prefix Information Options with very short lifetimes, on the premise these packets represented a bigger risk than other ND-based attack vectors [IPNG-minutes].

While reconsidering the trade-offs represented by such decision, we conclude that the drawbacks of mitigating the aforementioned attack vector outweigh the possible benefits.

In scenarios where RA-based attacks are of concern, proper mitigations such as RA-Guard [RFC6105] [RFC7113] or SEND [RFC3971] should be implemented.

4.3. Interface Initialization

When an interface is initialized, it is paramount that network configuration information is spread on the corresponding network (particularly in scenarios where an interface has been re-initialized, and the conveyed information has changed). Thus, this document replaces the following text from Section 6.2.4 of [RFC4861]:

In such cases, the router MAY transmit up to MAX_INITIAL_RTR_ADVERTISEMENTS unsolicited advertisements, using the same rules as when an interface becomes an advertising interface.

with:

In such cases, the router SHOULD transmit MAX_INITIAL_RTR_ADVERTISEMENTS unsolicited advertisements, using the same rules as when an interface becomes an advertising interface.

RATIONALE:
* Use of stale information can lead to interoperability problems. Therefore, it is important that new configuration information propagates in a timelier manner to all hosts.

NOTE:
[RFC9096] specifies recommendations for CPE routers to deprecate any stale network configuration information.
4.4. Conveying Information in Router Advertisement (RA) Messages

[TBD]

4.5. Recovery from Stale Configuration Information without Explicit Signaling

[TBD]

5. IANA Considerations

This document has no actions for IANA.

6. Implementation Status

[NOTE: This section is to be removed by the RFC-Editor before this document is published as an RFC.]

This section summarizes the implementation status of the updates proposed in this document. In some cases, they correspond to variants of the mitigations proposed in this document (e.g., use of reduced default lifetimes for PIOs, albeit using different values than those recommended in this document). In such cases, we believe these implementations signal the intent to deal with the problems described in [RFC8978] while lacking any guidance on the best possible approach to do it.

6.1. More Appropriate Lifetime Values

6.1.1. Router Configuration Variables

6.1.1.1. rad(8)

We have produced a patch for OpenBSD’s rad(8) [rad] that employs the default lifetimes recommended in this document, albeit it has not yet been committed to the tree. The patch is available at: <https://www.gont.com.ar/code/fgont-patch-rad-pio-lifetimes.txt>.

6.1.1.2. radvd(8)

The radvd(8) daemon [radvd], normally employed by Linux-based router implementations, currently employs different default lifetimes than those recommended in [RFC4861]. radvd(8) employs the following default values [radvd.conf]:

* Preferred Lifetime: 14400 seconds (4 hours)
* Valid Lifetime: 86400 seconds (1 day)
This is not following the specific recommendation in this document, but is already a deviation from the current standards.

6.2. Honor Small PIO Valid Lifetimes

6.2.1. Linux Kernel

A Linux kernel implementation of this document has been committed to the net-next tree. The implementation was produced in April 2020 by Fernando Gont <fgont@si6networks.com>. The corresponding patch can be found at: <https://patchwork.ozlabs.org/project/netdev/patch/20200419122457.GA971@archlinux-current.localdomain/>

6.2.2. NetworkManager

NetworkManager processes RA messages with a Valid Lifetime smaller than two hours as recommended in this document.

6.3. Conveying Information in Router Advertisement (RA) Messages

We know of no implementation that splits network configuration information into multiple RA messages.

6.4. Recovery from Stale Configuration Information without Explicit Signaling

6.4.1. dhcpcd(8)

The dhcpcd(8) daemon [dhcpcd], a user-space SLAAC implementation employed by some Linux-based and BSD-derived operating systems, will set the Preferred Lifetime of addresses corresponding to a given prefix to 0 when a single RA from the router that previously advertised the prefix fails to advertise the corresponding prefix. However, it does not affect the corresponding Valid Lifetime. Therefore, it can be considered a partial implementation of this feature.

6.5. Other mitigations implemented in products

[FRITZ] is a Customer Edge Router that tries to deprecate stale prefixes by advertising stale prefixes with a Preferred Lifetime of 0, and a Valid Lifetime of 2 hours (or less). There are two things to note with respect to this implementation:

* Rather than recording prefixes on stable storage (as recommended in [RFC9096]), this implementation checks the source address of IPv6 packets, and assumes that usage of any address that does not correspond to a prefix currently-advertised by the Customer Edge
Router is the result of stale network configuration information. Hence, upon receipt of a packet that employs a source address that does not correspond to a currently-advertised prefix, this implementation will start advertising the corresponding prefix with small lifetimes, with the intent of deprecating it.

* Possibly as a result of item "e)" (pp. 19-20) from Section 5.5.3 of [RFC4862] (discussed in Section 4.2 of this document), upon first occurrence of a stale prefix, this implementation will employ a decreasing Valid Lifetime, starting from 2 hours (7200 seconds), as opposed to a Valid Lifetime of 0.

7. Security Considerations

The protocol update in Section 4.2 could allow an on-link attacker to perform a Denial of Service attack against local hosts, by sending a forged RA with a PIO with a Valid Lifetime of 0. Upon receipt of that packet, local hosts would invalidate the corresponding prefix, and therefore remove any addresses configured for that prefix, possibly terminating e.g. TCP connections employing such addresses. However, an attacker may achieve similar effects via a number for ND-based attack vectors, such as directing traffic to a non-existing node until ongoing TCP connections time out, or performing a ND-based man-in-the-middle (MITM) attack and subsequently forging TCP RST segments to cause on-going TCP connections to be aborted. Thus, for all practical purposes, this attack vector does not really represent a greater risk than other ND attack vectors. As noted in Section 4.2, in scenarios where RA-based attacks are of concern, proper mitigations such as RA-Guard [RFC6105] [RFC7113] or SEND [RFC3971] should be implemented.

8. Acknowledgments

The authors would like to thank (in alphabetical order) Mikael Abrahamsson, Tore Anderson, Luis Balbinot, Brian Carpenter, Lorenzo Colitti, Owen DeLong, Gert Doering, Thomas Haller, Nick Hilliard, Bob Hinden, Philip Homburg, Lee Howard, Christian Huitema, Tatuya Jinmei, Erik Kline, Ted Lemon, Jen Linkova, Albert Manfredi, Roy Marples, Florian Obser, Jordi Palet Martinez, Michael Richardson, Hiroki Sato, Mark Smith, Hannes Frederic Sowa, Dave Thaler, Tarko Tikan, Ole Troan, Eduard Vasilenko, and Loganaden Velvindron, for providing valuable comments on earlier versions of this document.

The algorithm specified in Section 4.5 is the result of mailing-list discussions over previous versions of this document with Philip Homburg.
Fernando would like to thank Alejandro D’Egidio and Sander Steffann for a discussion of these issues, which led to the publication of [RFC8978], and eventually to this document.

Fernando would also like to thank Brian Carpenter who, over the years, has answered many questions and provided valuable comments that has benefited his protocol-related work.

9. References

9.1. Normative References


9.2. Informative References

[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,

[dhcpcd]   Marples, R., "dhcpcd - a DHCP client",

(updated with solution)", SI6 Networks Blog, February

[IPNG-GTF] IETF, "IPNG working group (ipngwg) Meeting Minutes",
Proceedings of the thirty-eight Internet Engineering Task

[NetworkManager] NetworkManager, "NetworkManager web site",
<https://wiki.gnome.org/Projects/NetworkManager>.

[rad]      Obser, F., "OpenBSD Router Advertisement Daemon - rad(8)",

Advertisement Daemon (radvd)",

file of the router advertisement daemon",


[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,
Appendix A.  Analysis of Some Suggested Workarounds

[This section is to be removed before publication of this document as an RFC].

During the discussion of this document, some alternative workarounds were suggested on the 6man mailing-list. The following subsections analyze these suggested workarounds, in the hopes of avoiding rehashing the same discussions.

Gont, et al.  Expires 8 December 2022
A.1. On a Possible Reaction to ICMPv6 Error Messages

It has been suggested that if configured addresses become stale, a CPE enforcing ingress/egress filtering (BCP38) ([RFC2827]) could send ICMPv6 Type 1 (Destination Unreachable) Code 5 (Source address failed ingress/egress policy) error messages to the sending node, and that, upon receipt of such error messages, the sending node could perform heuristics that might help to mitigate the problem discussed in this document.

The aforementioned proposal has a number of drawbacks and limitations:

* It assumes that the CPE routers enforce ingress/egress filtering [RFC2827]. While this is desirable behaviour, it cannot be relied upon.

* It assumes that if the CPE enforces ingress/egress filtering, the CPE will signal the packet drops to the sending node with ICMPv6 Type 1 (Destination Unreachable) Code 5 (Source address failed ingress/egress policy) error messages. While this may be desirable, [RFC2827] does not suggest signaling the packet drops with ICMPv6 error messages, let alone the use of specific error messages (such as Type 1 Code 5) as suggested.

* ICMPv6 Type 1 Code 5 could be interpreted as the employed address being stale, but also as a selected route being inappropriate/suboptimal. If the later, deprecating addresses or invalidating addresses upon receipt of these error messages would be inappropriate.

* Reacting to these error messages would create a new attack vector that could be exploited from remote networks. This is of particular concern since ICMP-based attacks do not even require that the Source Address of the attack packets be spoofed [RFC5927].

A.2. On a Possible Improvement to Source Address Selection

[RFC6724] specifies source address selection (SAS) for IPv6. Conceptually, it sorts the candidate set of source addresses for a given destination, based on a number of pair-wise comparison rules that must be successively applied until there is a "winning" address.

An implementation might improve source address selection, and prefer the most-recently advertised information. In order to incorporate the "freshness" of information in source address selection, an implementation would be updated as follows:
The node is assumed to maintain a timer/counter that is updated at least once per second. For example, the time(2) function from unix-like systems could be employed for this purpose.

The local information associated with each prefix advertised via RAs on the local network is augmented with a "LastAdvertised" timestamp value. Whenever an RA with a PIO with the "A" bit set for such prefix is received, the "LastAdvertised" timestamp is updated with the current value of the timer/counter.

[RFC6724] is updated such that this rule is incorporated:

Rule 7.5: Prefer fresh information  If one of the two source addresses corresponds to a prefix that has been more recently advertised, say LastAdvertised(SA) > LastAdvertised(SA), then prefer that address (SA in our case).

A clear benefit of this approach is that a host will normally prefer "fresh" addresses over possibly stale addresses.

However, there are a number of drawbacks associated with this approach:

* In scenarios where multiple prefixes are being advertised on the same LAN segment, the new SAS rule is *guaranteed* to result in non-deterministic behaviour, with hosts frequently changing the default source address. This is certainly not desirable from a troubleshooting perspective.

* Since the rule must be incorporated before "Rule 8: Use longest matching prefix" from [RFC6724], it may lead to suboptimal paths.

* This new rule may help to improve the selection of a source address, but it does not help with the housekeeping (garbage collection) of configured information:
  
  - If the stale prefix is re-used in another network, nodes employing stale addresses and routes for this prefix will be unable to communicate with the new "owner" of the prefix, since the stale prefix will most likely be considered "on-link".
  
  - Given that the currently recommended default value for the "Valid Lifetime" of PIOs is 2592000 seconds (30 days), it would take too long for hosts to remove the configured addresses and routes for the stale prefix. While the proposed update in Section 4.1 of this document would mitigate this problem, the lifetimes advertised by the local SLAAC router are not under the control of hosts.
As a result, updating IPv6 source address selection does not relieve nodes from improving their SLAAC implementations as specified in Section 4, if at all desirable. On the other hand, the algorithm specified in Section 4.5 would result in Rule 3 of [RFC6724] employing fresh addresses, without leading to non-deterministic behaviour.

Authors’ Addresses

Fernando Gont  
SI6 Networks  
Segurola y Habana 4310, 7mo Piso  
Villa Devoto  
Ciudad Autonoma de Buenos Aires  
Argentina  
Email: fgont@si6networks.com  
URI: https://www.si6networks.com

Jan Zorz  
6connect  
Email: jan@connect.com

Richard Patterson  
Sky UK  
Email: richard.patterson@sky.uk
This document proposes a new Hop-by-Hop option of IPv6 extension header to carry the topology identifier, which is used to identify the forwarding table instance created by the Multi Topology Routing or Flexible Algorithm.

Requirements Language

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at 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 September 22, 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) in effect on the date of
1. Introduction

This document proposes a new Hop-by-Hop option of IPv6 extension header to carry the topology identifier, which is used to identify the forwarding table instance created by the Multi Topology Routing or Flexible Algorithm.

2. Terminologies

The following terminologies are used in this document.

MT: Multi Topology

3. Problem Statement

[RFC4915] defines Multi-Topology Routing in OSPF and [RFC5120] defines Multi Topology Routing in ISIS. Through Multi Topology Routing, there can be multiple forwarding tables in the data plane. [I-D.ietf-lsr-flex-algo] can also implement the similar purpose to install multiple forwarding tables for different Flexible Algorithm instances.

In the MT IP Forwarding Considerations of [RFC5120], it is explained when multiple MTs share an Interface with overlapping addresses, some additional mechanism is needed to select the correct RIBs for the incoming IP packets to determine the correct RIB to make a forwarding decision. But there is lack of the generic approach of packet to multiple MT RIB mapping over the same inbound interface.
4. New IPv6 Extension Header Option for Topologies

In order to solve the above issue, in the scenario of IPv6, the topology identifier information can be carried in the packet. A new Hop-by-Hop option type "Topology" is defined to carry the topology related Identifier in an IPv6 packet. Its format is shown as below:

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Opt Type   |    Opt Data Len     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Topology ID        |   Reserved          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Figure 1. Topology Option
```

where:

- Opt Type: Type value is TBD. 8-bit unsigned integer. Identifier of the type of this Topology Option.

- Opt Data Len: 8-bit unsigned integer. It indicates the length of the option Data field of this option, in octets. The value of Opt Data Len of Topology option SHOULD be set to 4.

- Topology ID: 2-octet identifier which uniquely identifies the topology associated with the specific forwarding table created by the Multi Topology Routing or Flexible Algorithm.

- Reserved: 2-octet reserved field. It MUST be set as 0 and ignored when received.

5. Security Considerations

TBD

6. IANA Considerations

This document requests IANA to assign a new option type from "Hop-by-Hop Options" registry.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>Topology Option</td>
<td>this document</td>
</tr>
</tbody>
</table>
7. References

7.1. Normative References

[I-D.ietf-lsr-flex-algo]


7.2. Informative References


Authors’ Addresses

Zhenbin Li
Huawei Technologies
Beijing 100095
China

Email: lizhenbin@huawei.com

Zhibo Hu
Huawei Technologies
Beijing 100095
China

Email: huzhibo@huawei.com
Jie Dong
Huawei Technologies
Beijing 100095
China

Email: jie.dong@huawei.com
Abstract

This document defines the generalized IPv6 tunnel based on the analysis of challenges of the existing problems of IP tunnels.

Requirements Language

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

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 12 January 2023.

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) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights
1. Introduction

Currently there are many types of IP tunnels, such as VXLAN and GRE. On IPv6 networks, it is hard to define extensions for all these tunnels to support new features. On the other hand it is not recommended to extend new features based on the IPv4 data plane for these tunnels [ ]. This document analyzes the problems of IP tunnels and defines a generalized IPv6 tunnel to support the new features.

2. Terminology

APN: Application-aware Networking
GRE: Generic Routing Encapsulation
IPv4: Internet Protocol version 4
IPv6: Internet Protocol version 6
IOAM: In-situ Operations, Administration, and Maintenance
ISATAP: Intra-Site Automatic Tunnel Addressing Protocol
L2TPv3: Layer Two Tunneling Protocol Version 3
MPLS: Multiprotocol Label Switching
NVO3: Network Virtualization Overlays

SRv6: Segment Routing IPv6

SID: SR Identification

VNI: VXLAN Network Identifier

VXLAN: Virtual eXtensible Local Area Network

3. Problem Statement

There have been many types of IP tunnels, including:
- GRE Tunnels: it is defined in [RFC2784].
- IP in IP Tunnels: it is defined in [RFC1853].
- L2TPv3 Tunnels: it is defined in [RFC3931].
- ISATAP Tunnels: it is defined in [RFC4214].
- IPv4/IPv6 over IPv6 (4over6) Tunnels: it is defined in [RFC2473].
- VXLAN Tunnels: it is defined in [RFC7348].
- NVGRE Tunnels: it is defined in [RFC7637].
- MPLS over UDP: it is defined in [RFC7510].
- VXLAN-GPE (Generic Protocol Extension for VXLAN) Tunnels: it is defined in [I-D.ietf-nvo3-vxlan-gpe].

Currently many new features are emerging and the corresponding encapsulations over the IPv6 are defined:
- [I-D.ietf-6man-ipv6-alt-mark] defines IPv6 encapsulation for Alternate Marking.
- [I-D.ietf-ippm-ioam-ipv6-options] defines IPv6 encapsulation for IOAM.
- [I-D.dong-6man-enhanced-vpn-vtn-id] defines the IPv6 encapsulation used to determine resource isolation.
- [I-D.li-apn-ipv6-encap] defines the IPv6 encapsulation of an APN.
If the existing IP tunnels need to support new features such as Alternate Marking, IOAM, resource isolation, and APN, the following problems exist:

- All of the IP tunnels mentioned above need to be extended accordingly, resulting in a lot of standardization work.

- It is hard to keep the consistency between IPv4 and IPv6 for these IP tunnels (except IPv4 transition tunnels) since the possible extensions are recommended to be only done over the IPv6.

- IPv6 can directly support some functions of these IP tunnels which cannot be done over the IPv4. This means such functions become redundant over the IPv6. For example, VXLAN takes use of the UDP to support ECMP. However for the IPv6 VXLAN, the Flow Label in the IPv6 header can also be used to support ECMP.

- Some IP tunnels such as VXLAN and GRE have their own headers. If these tunnels need to support new features over the IPv6, there will face the challenge of the choice between reusing the exiting IPv6 encapsulations for these new features based on the IPv6 extension header and define new extensions based on their own tunnel headers.

  -- If the tunnel header is extended, it will be redundant with the existing IPv6 encapsulation for the new features based on the IPv6 extension header.

  -- For some existing IP tunnels (such as IP in IP) that do not have their own headers, they have to reuse the IPv6 encapsulations for these new features based the IPv6 header. extensions need to be redefined in the IPv6 extension header. As a result, their extensions may be different from that of the IP tunnels which have their own headers.

4. Design Consideration

In order to solve the above problems, the following choice can be taken into account:

1. It is not recommended to support the new features over the IPv4 for these IP tunnels.

2. It is to reuse the existing IPv6 encapsulations based on the IPv6 extension header when support the new features for these IP tunnels over the IPv6.
If these IP tunnels take use of the existing encapsulations based on the IPv6 extension header, for the IP tunnels which have their own headers, there are two possible options to cope with their own headers:

Option 1: They use the IPv6 extension headers to support new features and their own headers are retained.

Option 2: Define a generalized IPv6 tunnel that contains the IPv6 extension header which not only reuses the existing IPv6 encapsulations to support new features, but also introduces the new extensions to support the necessary functions of their own headers.

But the Option 1 has the following problems:

1. Redundant Functions: For example, the UDP-based IP tunnel can directly use IPv6 flow label to implement ECMP.

2. Separated process of tunnel functions: Some functions use IPv6 extension headers and some functions use headers of tunnels. As a result, tunnel processing is separated and complex. If the IPSec extension header is used, the tunnel’s own header maybe encrypted and unable to be parsed when necessary process is needed.

According to the above design consideration, Option 2 is recommended in this document.

5. Structure of a GIP6 Encapsulated Packet

The Generalized IPv6 (GIP6) tunnel is defined to use the IPv6 header and IPv6 extension header to support both existing IP tunnels functions and new features.

A GIP6 encapsulated packet has the following format:

```
+---------------+           +---------------+
| IPv6 Header   |           | GIP6 Encapsulation |
+---------------+           +---------------+
| IPv6 Extension Header |
| (Encapsulations of new features + Encapsulations of functions of existing IP tunnels) |
+---------------+           +---------------+
| Payload packet |
+---------------+
```

Figure 1. GIP6 Encapsulation
Different types of tunneled payloads, such as IPv4, IPv6, MPLS, Ethernet, etc., can be indicated by the IPv6 Next Header.

Packets encapsulated with the GIP6 tunnel can directly support new functions based on existing encapsulation, including Alternate-Marking, IOAM, Resource-Isolation, and APN.

6. GIP6 for VXLAN

To support existing VXLAN functions, the GIP6 tunnel is extended as follows:

1. The function of the UDP is replaced by the flow label of the IPv6 header in the GIP6 tunnel. To ensure compatibility, the value of the flow label calculated for the purpose of ECMP SHOULD be the same as that of the source port of the UDP.

2. Definition of the VN Option

A new option called VN Option is defined to carry the VXLAN header information. The VN Option MUST only be encapsulated in the Destination Options Header (DOH).

The following figure shows the data fields format of the VN option:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Option Type | Opt Data Len |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| VXLAN Header (8 Bytes) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2. VN Option

The VN Option contains the following fields:

* Option Type: 8-bit selector. VN option. Value TBD by IANA.

* 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 8.

* Option Data: 64-bits VXLAN Header Information.
The following figure shows the definition of VXLAN headers in [RFC7358]. For the detailed definition of the data fields, please refer to [RFC7358].

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| VXLAN Flags |                  Reserved                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       VXLAN Network Identifier(VNI)           |  Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3. VXLAN headers

7. GIPv6 for GRE

A new option called GRE Option is defined to carry the GRE header information. The GRE Option MUST only be encapsulated in the Destination Options header (DOH).

The definition of a new TLV for the Options Extension Headers, carrying the data fields dedicated to the GRE information, is reported below.

The following figure shows the data fields format of the GRE option.

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Option Type  |  Opt Data Len |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                             |
|GRE Header (16 Bytes)                                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4. GRE Option

The GRE Option contains the following fields:

* Option Type: 8-bit selector. GRE option. Value TBD by IANA.

* 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 16.

* Option Data: 128-bits GRE Header Information.
The following figure shows the definition of the GRE header in [RFC2890]. For the detailed definition of the data fields, please refer to [RFC2784] and [RFC2890].

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C| |K|S| Reserved0       | Ver |         Protocol Type         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Checksum (optional)      |       Reserved1 (Optional)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Key (optional)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Sequence Number (Optional)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Figure 5. GRE Header
```

Note: The function of the Protocol Type in the GRE header can be replaced by that of the NH in the IPv6 header or IPv6 extension header.

8. GIP6 for Other Existing IP Tunnels

They will be defined in the future version.

9. Security Considerations

TBD.

10. IANA Considerations

The Option Type should be assigned in IANA’s "Destination Options" registry.

This draft requests the following IPv6 Option Type assignment from the Destination Options sub-registry of Internet Protocol Version 6 (IPv6) Parameters (https://www.iana.org/assignments/ipv6-parameters/).

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Binary Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>00</td>
<td>0</td>
<td>TBD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VN</td>
<td>[This draft]</td>
</tr>
<tr>
<td>TBD</td>
<td>00</td>
<td>0</td>
<td>TBD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRE</td>
<td>[This draft]</td>
</tr>
</tbody>
</table>

Figure 6. IANA Considerations
11. References

[I-D.dong-6man-enhanced-vpn-vtn-id]
Dong, J., Li, Z., Xie, C., Ma, C., and G. Mishra,
"Carrying Virtual Transport Network (VTN) Identifier in
IPv6 Extension Header", Work in Progress, Internet-Draft,
draft-dong-6man-enhanced-vpn-vtn-id-06, 24 October 2021,

[I-D.ietf-6man-ipv6-alt-mark]
Fioccola, G., Zhou, T., Cociglio, M., Qin, F., and R.
Pang, "IPv6 Application of the Alternate Marking Method",
Work in Progress, Internet-Draft, draft-ietf-6man-ipv6-alt-mark-16, 1 July 2022,

[I-D.ietf-ippm-ioam-ipv6-options]
Bhandari, S. and F. Brockners, "In-situ OAM IPv6 Options",
Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-ipv6-options-08, 16 June 2022,

[I-D.ietf-nvo3-vxlan-gpe]
(Editor), F. M., (editor), L. K., and U. E. (editor),
"Generic Protocol Extension for VXLAN (VXLAN-GPE)",
Work in Progress, Internet-Draft, draft-ietf-nvo3-vxlan-gpe-12,

[I-D.li-apn-ipv6-encap]
Li, Z., Peng, S., and C. Xie, "Application-aware IPv6
Networking (APN6) Encapsulation",
Work in Progress, Internet-Draft, draft-li-apn-ipv6-encap-05, 31 May 2022,

[RFC1853] Simpson, W., "IP in IP Tunneling", RFC 1853,
DOI 10.17487/RFC1853, October 1995,

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,


Authors' Addresses

Li, et al. Expires 12 January 2023
Zhenbin
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China
Email: lizhenbin@huawei.com

Shuanglong
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China
Email: chenshuanglong@huawei.com

Haibo
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China
Email: rainsword.wang@huawei.com
Export of Segment Routing IPv6 Information in
IP Flow Information Export (IPFIX)
draft-tgraf-opsawg-ipfix-srv6-srh-05

Abstract

This document introduces new IP Flow Information Export (IPFIX)
information elements to identify the SRv6 Segment Routing Header
dimensions, the SRv6 Control Plane Protocol and the SRv6 Endpoint
Behavior that traffic is being forwarded with.

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 25 January 2023.

Copyright Notice

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

A new type of Routing Extension Header called Segment Routing Header (SRH) is defined by [RFC8754] which is used for applying Segment Routing (SR) on the IPv6 data plane.
Three routing protocol extensions, OSPFv3 Extensions
[I-D.li-lsr-ospfv3-srv6-extensions], IS-IS Extensions
[I-D.ietf-li-lsr-isis-srv6-extensions], BGP Prefix Segment Identifiers
(Prefix-SIDs) [I-D.ietf-bess-srv6-services] and one Path Computation
Element Communication Protocol (PCEP) Extension
[I-D.ietf-pce-segment-routing-ipv6] have been defined to propagate
Segment Identifiers (SIDs) for the IPv6 data plane.

SRv6 segment endpoint behaviors have been defined in [RFC8986] and
describe how packets should be processed.

This document defines eleven new IPFIX Information Elements (IEs) and
tree new subregistries within the "IPFIX Information Elements"
registry [RFC7012], respectively for the new SRH dimensions, SRv6
endpoint behaviors and routing protocol and PCEP extensions.

2. IPFIX Information Elements

This section defines and describes the new IPFIX IEs.

srhFlagsIPv6
8-bit flags defined in the SRH.

srhTagIPv6
16-bit tag field defined in the SRH that marks a packet as part of
a class or group of packets sharing the same set of properties.

srhSegmentIPv6
128-bit IPv6 address that represents an SRv6 segment.

srhActiveSegmentIPv6
128-bit IPv6 address that represents the active SRv6 segment.

srhSegmentIPv6BasicList
Ordered basicList [RFC6313] of zero or more 128-bit IPv6 addresses
in the SRH that represents the SRv6 segment list. The Segment
List is encoded starting from the last segment of the SR Policy.
That is, 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.

srhSegmentIPv6ListSection
Exposes the SRH Segment List as defined in section 2 of [RFC8754]
as series of n octets.

srhSegmentIPv6sLeft
8-bit unsigned integer defining the number of route segments
remaining to reach the end of the segment list.
srhSectionIPv6
Exposes the SRH and its TLV’s as defined in section 2 of [RFC8754] as series of n octets.

srhActiveSegmentIPv6Type
Name of the routing protocol or PCEP extension from where the active SRv6 segment has been learned from.

srhSegmentLocatorLength
The number of significant bits. Together with srhSegmentIPv6 it enables the calculation of the SRv6 Locator.

srhSegmentEndpointBehavior
16-bit unsigned integer that represents a SRv6 Endpoint behavior.

Note that the srhSegmentIPv6, srhSegmentLocatorLength, and srhSegmentEndpointBehavior IPFIX IEs are generic fields, to be used in the context of IPFIX Options Templates or IPFIX Structured Data [RFC6313].

3. Use Cases
By using srhSegmentIPv6BasicList(TBD5) or the srhSegmentIPv6ListSection (TBD6), srhActiveSegmentIPv6 (TBD4), srhSegmentIPv6sLeft (TBD7), srhActiveSegmentIPv6Type(TBD9), the forwardingStatus(89), and some counters information, it is possible to answer the following questions (amongst others):

- how many packets are forwarded or dropped
- if dropped, for which reasons,
- identify the active segment and its control plane protocol,
- the SRv6 segment list,
- the next SRv6 node and its type,
- and how many SRv6 segments are left.

4. IANA Considerations
This document requests IANA to create new IEs (see table 1) and three new subregistries called "IPFIX IPv6 SRH Flags" (table 2), "IPFIX IPv6 SRH Segment type" (table 3) and "IPFIX SRV6 Endpoint Behavior" (table 4) under the "IPFIX Information Elements" registry [RFC7012] available at [IANA-IPFIX] and assign the following initial code points.
<table>
<thead>
<tr>
<th>Element ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD1</td>
<td>srhFlagsIPv6</td>
</tr>
<tr>
<td>TBD2</td>
<td>srhTagIPv6</td>
</tr>
<tr>
<td>TBD3</td>
<td>srhSegmentIPv6</td>
</tr>
<tr>
<td>TBD4</td>
<td>srhActiveSegmentIPv6</td>
</tr>
<tr>
<td>TBD5</td>
<td>srhSegmentIPv6BasicList</td>
</tr>
<tr>
<td>TBD6</td>
<td>srhSegmentIPv6ListSection</td>
</tr>
<tr>
<td>TBD7</td>
<td>srhSegmentIPv6sLeft</td>
</tr>
<tr>
<td>TBD8</td>
<td>srhSectionIPv6</td>
</tr>
<tr>
<td>TBD9</td>
<td>srhActiveSegmentIPv6Type</td>
</tr>
<tr>
<td>TBD10</td>
<td>srhSegmentLocatorLength</td>
</tr>
<tr>
<td>TBD11</td>
<td>srhSegmentEndpointBehavior</td>
</tr>
</tbody>
</table>

Table 1: Creates IEs in the "IPFIX Information Elements" registry

Note to the RFC-Editor:

* Please replace TBD1 - TBD16 with the values allocated by IANA
* Please replace the [RFC-to-be] with the RFC number assigned to this document
4.1. srhFlagsIPv6

Name: srhFlagsIPv6 ElementID: TBD1 Description: This Information Element identifies the 8-bit flags defined in the SRH. Values for this Information Element are listed in the "IPFIX IPv6 SRH Flags" registry, see Abstract Data Type: unsigned8 Data Type Semantics: flags Reference: [RFC-to-be], RFC8754[IANA-IPFIX]. srhFlagsIPv6 values must not be directly added to this "IPFIX IPv6 SRH Flags" registry. They must instead be added to the "Segment Routing Header Flags" registry. Both the "IPFIX IPv6 SRH Flags" and the "Segment Routing Header Flags" registries must be kept in synch. Initial values in the registry are defined by the table below.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>O-flag</td>
<td>[RFC-ietf-6man-spring-srv6-oam-13]</td>
</tr>
<tr>
<td>3-7</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: "IPFIX IPv6 SRH Flags" registry

4.2. srhTagIPv6

Name: srhTagIPv6 ElementID: TBD2 Description: This Information Element identifies the 16-bit tag field defined in the SRH that marks a packet as part of a class or group of packets sharing the same set of properties. Abstract Data Type: unsigned16 Data Type Semantics: identifier Reference: [RFC-to-be], RFC8754

4.3. srhSegmentIPv6

Name: srhSegmentIPv6 ElementID: TBD3 Description: This Information Element identifies the 128-bit IPv6 address that represents an SRv6 segment. Abstract Data Type: ipv6address Data Type Semantics: default Reference: [RFC-to-be], RFC8754

4.4. srhActiveSegmentIPv6

Name: srhActiveSegmentIPv6 ElementID: TBD4 Description: This Information Element identifies the 128-bit IPv6 address that represents the active SRv6 segment. Abstract Data Type: ipv6address Data Type Semantics: default Reference: [RFC-to-be], RFC8754
4.5. srhSegmentIPv6BasicList

Name: srhSegmentIPv6BasicList ElementID: TBD5 Description: This Information Element identifies the Ordered basicList [RFC6313] of zero or more 128-bit IPv6 addresses in the SRH that represents the SRv6 segment list. The Segment List is encoded starting from the last segment of the SR Policy. That is, 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. Abstract Data Type: basicList Data Type Semantics: list Reference: [RFC-to-be], RFC8754

4.6. srhSegmentIPv6ListSection

Name: srhSegmentIPv6ListSection ElementID: TBD6 Description: Exposes the SRH Segment List as defined in section 2 of Abstract Data Type: octetArray Data Type Semantics: default Reference: [RFC-to-be], RFC8754 as series of n octets.

4.7. srhSegmentIPv6sLeft

Name: srhSegmentIPv6sLeft ElementID: TBD7 Description: This Information Element identifies the 8-bit unsigned integer defining the number of route segments remaining to reach the end of the segment list. Abstract Data Type: unsigned8 Data Type Semantics: quantity Reference: [RFC-to-be], RFC8754

4.8. srhSectionIPv6

Name: srhSectionIPv6 ElementID: TBD8 Description: This Information Element exposes the SRH and its TLV’s as defined in section 2 of Abstract Data Type: octetArray Data Type Semantics: default Reference: [RFC-to-be], RFC8754 as series of n octets.

4.9. srhActiveSegmentIPv6Type

Name: srhActiveSegmentIPv6Type ElementID: TBD9 Description: This Information Element identifies the name of the routing protocol or PCEP extension from where the active SRv6 segment has been learned from. Values for this Information Element are listed in the "IPFIX IPv6 SRH Segment type" registry, see Abstract Data Type: unsigned8 Data Type Semantics: identifier Reference: [RFC-to-be][IANA-IPFIX]. Initial values in the registry are defined by the table below. New assignments of values will be administered by IANA and are subject to Expert Review [RFC8126]. Experts need to check definitions of new values for completeness, accuracy, and redundancy.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD12</td>
<td>Unknown</td>
<td>[RFC-to-be]</td>
</tr>
<tr>
<td>TBD13</td>
<td>Path Computation Element</td>
<td>[RFC-to-be], draft-ietf-pce-segment-routing-ipv6</td>
</tr>
<tr>
<td>TBD14</td>
<td>OSPFv3 Segment Routing</td>
<td>[RFC-to-be], draft-li-ospf-ospfv3-srv6-extensions</td>
</tr>
<tr>
<td>TBD15</td>
<td>IS-IS Segment Routing</td>
<td>[RFC-to-be], draft-ietf-lsr-isis-srv6-extensions</td>
</tr>
<tr>
<td>TBD16</td>
<td>BGP Segment Routing Prefix-SID</td>
<td>[RFC-to-be], draft-ietf-bess-srv6-services</td>
</tr>
</tbody>
</table>

Table 3: "IPFIX IPv6 SRH Segment type" subregistry

4.10. srhSegmentLocatorLength

Name: srhSegmentLocatorLength  ElementID: TBD10  Description: This Information Element identifies the number of significant bits and together with srhSegmentIPv6 enables the calculation of the SRv6 Locator. Abstract Data Type: unsigned8  Data Type Semantics: default  Reference: [RFC-to-be], RFC8986 Section 3.1

4.11. srhSegmentEndpointBehavior

Name: srhSegmentEndpointBehavior  ElementID: TBD11  Description: This Information Element identifies the 16-bit SRv6 Endpoint behavior. Values for this Information Element are listed in the "IPFIX SRv6 Endpoint Behavior" registry, see Abstract Data Type: unsigned16  Data Type Semantics: identifier  Reference: [RFC-to-be], RFC8986 Section 4[IANA-IPFIX]. srhSegmentEndpointBehavior values must not be directly added to this "IPFIX SRv6 Endpoint Behavior" registry. They must instead be added to the "Segment Routing SRv6 Endpoint Behaviors" registry. Both the "IPFIX SRv6 Endpoint Behavior" and the "Segment Routing SRv6 Endpoint Behaviors" registries must be kept in synch.
Table 4: "IPFIX SRV6 Endpoint Behavior" registry

5. Operational Considerations

5.1. SRv6 Segment List

The zero or more 128-bit IPv6 addresses in the SRH [RFC8754] can be exported in two different ways, with two different IPFIX IEs:

* srhSegmentIPv6BasicList
* srhSegmentIPv6ListSection

The srhSegmentIPv6BasicList encodes the SID list of IPv6 addresses with a basicList, specified in the IPFIX Structured Data [RFC6313]. This encoding offers the advantage to the data collection that the different IPv6 addresses are already structured as a list, without the need of post processing. However, this method requires some extra processing on the exporter, to realize the BasicList data mapping.

The srhSegmentIPv6ListSection, on the other hand, encodes the list of IPv6 addresses as an octetArray. This doesn't impose any data flow manipulation on the exporter, facilitating the immediate export. However, the data collection must be able to decode the IPv6 addresses according the SR specifications. Compared to the srhSegmentIPv6BasicList, the srhSegmentIPv6ListSection flow records length is slightly reduced.

It is not expected that an exporter would support both srhSegmentIPv6BasicList and srhSegmentIPv6ListSection at the same time.
5.2. Compressed SRv6 Segment List Decomposition

The SRv6 segment list in the IPFIX IEs srhSegmentIPv6BasicList and srhSegmentIPv6ListSection could contain compressed-SID containers as described in [I-D.ietf-spring-srv6-srh-compression]. The SID endpoint behaviors described in section 4 of [I-D.ietf-spring-srv6-srh-compression] determine wherever the segment list is compressed or not. The SID Locator as described in section 3.1 [RFC8986], determines the common most significant bits.

6. Security Considerations

There exists no significant extra security considerations regarding the allocation of these new IPFIX IEs compared to [RFC7012].

7. Acknowledgements

The authors would like to thank Yao Liu, Paolo Lucente, Eduard Vasilenko, Alex Huang Feng and Bruno Decraene for their review and valuable comments.

8. References

8.1. Normative References


8.2. Informative References
[I-D.ietf-bess-srv6-services]

[I-D.ietf-lsr-isis-srv6-extensions]

[I-D.ietf-pce-segment-routing-ipv6]

[I-D.ietf-spring-srv6-srh-compression]

[I-D.li-lsr-ospfv3-srv6-extensions]

[IANA-IPFIX]
Appendix A.  IPFIX Encoding Examples

This appendix represents three different encodings for the newly introduced IEs, for the example values in the table 5. The three different encodings uses the following IEs, respectively: srhSegmentIPv6BasicList, srhSegmentIPv6ListSection, and srhSectionIPv6.

<table>
<thead>
<tr>
<th>SRH Nr</th>
<th>SRH Flags</th>
<th>SRH Tag</th>
<th>Active Segment Type</th>
<th>Segment List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>123</td>
<td>IS-IS [TBD15]</td>
<td>2001:db8::1, 2001:db8::2, 2001:db8::3</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>456</td>
<td>IS-IS [TBD15]</td>
<td>2001:db8::4, 2001:db8::5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>789</td>
<td>IS-IS [TBD15]</td>
<td>2001:db8::6</td>
</tr>
</tbody>
</table>

Table 5: three observed SRH headers and their routing protocol

A.1.  Template Record and Data Set with Segment Basic List

With this encoding, the examples in Table 5 are represented with the following IEs:

* SR Flags => srhFlagsIPv6
* SR Tag => srhTagIPv6
* Active Segment Type => srhActiveSegmentIPv6Type
* Segment List => srhSegmentIPv6BasicList
Table 6: Template Record with Basic List Encoding Format

In this example, the Template ID is 256, which will be used in the Data Record. The field length for srhSegmentIPv6BasicList is 0xFFFF, which means the length of this IE is variable, and the actual length of this IE is indicated by the List Length field in the basicList format as per [RFC6313].

The data set is represented as follows:

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         SET ID = 256          |           Length = 136        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| srhFlagsIPv6  |        srhTagIPv6 = 123       | srhActiveSegmentIPv... = TBD15|
| = 0           |                               |                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      255      |        List Length = 53       | semantic= ordered |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     srhSegmentIPv6 = TBD3     |        Field Length = 16      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Segment List[0] = 2001:db8::1                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ...                                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ...                                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ...                                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Segment List[1] = 2001:db8::2

|-------------------------------|

srhFlagsIPv6 | srhTagIPv6 = 456 | srhActiveSegmentIPv6...
= 0 | semantic=ordered

255 | List Length = 37 |

srhSegmentIPv6 = TBD3 | Field Length = 16

Segment List[0] = 2001:db8::4

<table>
<thead>
<tr>
<th>Segment List[1] = 2001:db8::5 (16 bytes)</th>
</tr>
</thead>
</table>

srhFlagsIPv6 | srhTagIPv6 = 789 | srhActiveSegmentIPv6...
= 0 | semantic=ordered

255 | List Length = 21 |

srhSegmentIPv6 = TBD3 | Field Length = 16
A.2. Template Record and Data Set with Segment List Section

With this encoding, the examples in Table 5 are represented with the following IEs:

* SR Flags => srhFlagsIPv6
* SR Tag => srhTagIPv6
* Active Segment Type => srhActiveSegmentIPv6Type
* Segment List => srhSegmentIPv6List

<table>
<thead>
<tr>
<th>SET ID = 2</th>
<th>Length = 24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template ID = 257</td>
<td>Field Count = 4</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>0</td>
<td>srhFlagsIPv6 = TBD1</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>0</td>
<td>srhTagIPv6 = TBD2</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>0</td>
<td>srhActiveSegmentIPv... = TBD9</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>0</td>
<td>srhSegmentIPv6List</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------</td>
</tr>
</tbody>
</table>

Table 8: Template Record with Segment List Section Encoding Format

In this example, the Template ID is 257, which will be used in the Data Record. The field length for srhSegmentIPv6ListSection is 0xFFFF, which means the length of this IE is variable.
The data set is represented as follows:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>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</td>
<td></td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SET ID = 257</td>
</tr>
<tr>
<td></td>
<td>srhFlagsIPv6</td>
</tr>
<tr>
<td></td>
<td>= 0</td>
</tr>
<tr>
<td></td>
<td>ntIPv...=TBD15</td>
</tr>
<tr>
<td></td>
<td>0xFFFF</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>= 0</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>srhActiveSegme</td>
</tr>
<tr>
<td></td>
<td>ntIPv...=TBD15</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>+-----------------------------------------------+------------------</td>
<td></td>
</tr>
</tbody>
</table>
Table 9: Data Set Encoding Format for Segment List Section

A.3. Template Record and Data Set with SRH Section

With this encoding, the examples in Table 5 are represented with the following IEs:

* SR Flags + SR Tag + Segment List => srhSectionIPv6

* Active Segment Type => srhActiveSegmentIPv6Type

```
| = 789         | srhActiveSegmentIPv6Type=TBD15 |
|               | srhFlagIPv6=0 srhTagIPv6... |
|               | IPv6Type=TBD15 |
|               | srhActiveSegmentIPv6Type= TBD9 |
|               | srhSectionIPv6 = TBD8 |
|               | Field Length = 0xFFFF |
|               | Length = 16 |
|               | Field Count = 2 |
|               | Field Length = 1 |
|               | Field Length = 0xFFFF |
```

Table 10: Template Record with SRH Section Encoding Format
In this example, the Template ID is 258, which will be used in the Data Record. The field length for srhSectionIPv6 is 0xFFFF, which means the length of this IE is variable.

The data set is represented as follows:

```
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         SET ID = 258          |           Length = (*)        |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        srhActiveSegmentIPv6Type = TBD15       |    0xFFFF     |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header   |  Hdr Ext Len  | Routing Type  | Segments Left |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Last Entry   |     Flags     |              Tag              |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Segment List[0] 2001:db8::1                   |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Segment List[1] 2001:db8::2                   |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Optional Type Length Value objects (variable)
<table>
<thead>
<tr>
<th>Segment List[0] 2001:db8::4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>Segment List[1] 2001:db8::5</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>Optional Type Length Value objects (variable)</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>srhActiveSegmentIPv6Type = TBD15</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>Next Header</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>Last Entry</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>Segment List[0] 2001:db8::6</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>Optional Type Length Value objects (variable)</td>
</tr>
</tbody>
</table>

Table 11: Data Set Encoding Format for SRH Section

(*) The Length must be calculated to include the optional Type Length Value objects.
A.4. Options Template Record and Data Set for SRv6 end point behavior and Locator Length

This appendix provides an SRv6 EndPoint Behavior Options Template example, for the values presented in Table 12. In the Options Template case, the srhEndPointIPv6 Information Element is a Scope field

<table>
<thead>
<tr>
<th>Entry Nr</th>
<th>SRH End Point IPv6</th>
<th>SRH End Point Behavior</th>
<th>SRH Segment Locator Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2001:db8::1</td>
<td>End [1]</td>
<td>48</td>
</tr>
<tr>
<td>2</td>
<td>2001:db8::4</td>
<td>End with NEXT-CSID [43]</td>
<td>48</td>
</tr>
<tr>
<td>3</td>
<td>2001:db8::6</td>
<td>End.DX6 [16]</td>
<td>48</td>
</tr>
</tbody>
</table>

Table 12: three observed SRv6 End Point Behaviors

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>+-----------------------------------------------+-----------+------------------------+-----------------+--------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set ID = 3</td>
<td>Length = 24</td>
<td></td>
</tr>
<tr>
<td>+-----------------------------------------------+-----------+------------------------+-----------------+--------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Template ID 259</td>
<td>Field Count = 3</td>
<td></td>
</tr>
<tr>
<td>+-----------------------------------------------+-----------+------------------------+-----------------+--------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scope Field Count = 1</td>
<td>srhSegmentIPv6 = TBD3</td>
<td></td>
</tr>
<tr>
<td>+-----------------------------------------------+-----------+------------------------+-----------------+--------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scope 1 Field Length = 4</td>
<td>srhSegmentEndpointBeh.. = TBD11</td>
<td></td>
</tr>
<tr>
<td>+-----------------------------------------------+-----------+------------------------+-----------------+--------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Field Length = 1</td>
<td>srhSegmentLocatorLength = TBD10</td>
<td></td>
</tr>
<tr>
<td>+-----------------------------------------------+-----------+------------------------+-----------------+--------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Field Length = 4</td>
<td>Padding</td>
<td></td>
</tr>
</tbody>
</table>

Table 13: Template Record with SRH Section Encoding Format

In this example, the Template ID is 259, which will be used in the Data Record.

The data set is represented as follows:
Table 14: Data Set Encoding Format for SRH Section

(*) The Length must be calculated to include the optional Type Length Value objects.

Authors’ Addresses

Thomas Graf
Swisscom
Binzring 17
CH-8045 Zurich
Switzerland
Email: thomas.graf@swisscom.com

Benoit Claise
Huawei
Email: benoit.claise@huawei.com

Pierre Francois
INSA-Lyon
Lyon
France
Email: pierre.francois@insa-lyon.fr
 Neighbor Discovery support for Multi-home Multi-prefix
 draft-vv-6man-nd-support-mhmp-00

Abstract

Multi-home Multi-prefix IPv6 environment is the norm for businesses
that need to have uplink resiliency.

For any considered destination, the MHMP challenge may be split into
3 sub-challenges (important to solve in the below order):
1) the host should choose the proper source address for the packet,
2) the host should choose the best default router as the next-hop,
3) site topology may be complicated and may need the source routing
through the site.

This draft is concerned with the solution for the first two problems
that need improvement for the ND (RFC 4861) SLAAC (RFC 4862) and
Default Address Selection (RFC 6724). The last problem is considered
as properly discussed by Multihoming in Enterprise (RFC 8678).

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 2021.
Table of Contents

1. Terminology and pre-requisite........................................2
2. Introduction...........................................................3
3. Solution for the case "equal prefixes"..............................5
4. Solution for the case "non-equal prefixes"........................6
5. Resolution for a not reachable provider............................8
6. Extensions of the existing standards...............................10
   6.1. Preference to choose source address before the next-hop..10
   6.2. Default router choice by host................................11
   6.3. Prefixes become dynamic.......................................11
   6.4. Default router announcement rules..........................13
   6.5. Clean orphaned prefixes after default router list change.13
7. Interoperability analysis............................................13
8. Security Considerations.............................................14
9. IANA Considerations................................................14
10. References....................................................................14
    10.1. Normative References.........................................14
    10.2. Informative References......................................15
11. Acknowledgments......................................................16

1. Terminology and pre-requisite

[Default Address], [ND], and [SLAAC] are pre-requisite to understanding this document.
The terms are inherited from these standards.

Additional terms:
2. Introduction

Businesses usually have multiple network connections with different service providers to guarantee network resiliency. Such a scenario is identified as Multi-Home Multi-Prefix (MHMP) and properly discussed in [MHMP], [MHMP Enterprise].

An MHMP site may have a complex architecture in [IPv6], potentially, with many links and routers (CPEs) connected to different carriers. CPEs may receive different Provider Aggregatable (PA) prefixes from the upstream carriers.

Hosts located in an MHMP environment may also have multiple different addresses assigned to their interfaces that come from multiple delegations (from different carriers).

This may create challenges when a host located in an MHMP environment wishes to communicate with a certain destination. Such a host typically receives the list of destination addresses from DNS sorted by [Default Address]. Knowing the destination address, a host proceeds with the selection of the other parameters necessary for sending the packet.

The desired destination address is the pre-request for the discussion of this document, as pointed out in [Default Address]. This document does not change the way how destination addresses are sorted by [Default Address] or [Happy Eyeballs] rather it analyzes the options once the destination address is selected:
1. choose next-hop first (current practice for clients that prevail in MHMP environment; servers probably use bind() to choose source address but the server side is not relevant to the discussion because it typically has Provider Independent addresses),

or

2. choose source address first
   (more optimal strategy recommended here).

Both need some corrections to [ND] and [Default Address] selection.

In some cases, these choices may create issues, in particular when a faulty situation occurs (e.g. network prefixes invalidity due to loss of connectivity or abrupt CPE reconfiguration). During such events, a host may find it difficult to establish communication with a destination if the proper next-hop or source address is not selected, for example, due to the packet filtering policies applied by the upstream carriers (for anti-spoofing).

So, overall the association between a destination address, the next-hop, and the source address in an MHMP environment may be challenging.

The MHMP challenge may be split into 3 sub-challenges (important to solve in the below order):

1) the host should choose the proper source address for the packet by [Default Address],

2) the host should choose the best default router as the next-hop [ND] (not strictly mandatory, may be fixed later by the source routing with some loss of efficiency),

3) despite the assumed good choice for the default router selection, site topology may be complicated and as a result, may need the source routing through the site anyway – see [MHMP Enterprise].

It is important to point out that the challenge of selecting the next-hop and source address exists for every desired destination.

There are two reminders before the discussion for solutions:
o [ND] section 6.3.6 recommends the next-hop choice between default routers in a round-robin style. Traffic policy or even reachability of particular resources through a particular default router is not considered at the [ND] level.

o [Default Address] section 7 (and a few other places) assumes that source address selection should happen after the next-hop (or interface) choice by [ND].

Before digging into the discussions, it is worth noticing that:

o This document has put NAT (including NPT) out of consideration. The attempt is here to get fully transparent E2E connectivity.

o This document assumes PA environment only, PI (Provider Independent) address space needs BGP connection to Carrier that does not create a problem to solve from the technology point of view but it creates an enormous problem of scalability for the Internet with tens of millions of routes.

It is possible to introduce one additional classification to separate what it is possible to implement now from what needs additional standardization efforts:

1. Case "equal prefixes": Announced prefixes are fully equal by scope and value, all resources interested for hosts could be reachable through any announced PA prefix. Additionally, traffic distribution between carriers could be non-predictable (no traffic engineering or policy).

2. Case "non-equal prefixes": Announced prefixes are not equal because (1) some resources could be accessed only through a particular prefix (for example "walled garden" of one carrier) or (2) it is desirable to have some policy for traffic distribution between PA prefixes (cost of traffic, delay, packet loss, jitter, proportional load, etc.).

3. Solution for the case "equal prefixes"

This use case is potentially possible to operate without any changes to standardization but it would be not optimal (because currently next-hop is chosen first) and would need additional functionality on routers and hosts anyway (rule 5.5 of [Default Address], [Conditional PIO], source routing). Let’s discuss the current standardization situation.
Case "equal prefixes" does not create any requirement on what prefix should be used for the source address. It is only needed that the source address would be chosen to be compatible with the next-hop that should be in the direction of the respective carrier. There are 4 potential scenarios possible in respect of the next-hop choice:

1. A single router on the link does not create a choice for the host in principle. If the site is complex (multi-hop) then the router itself may need source routing to choose next-hop properly, it is considered resolved and properly discussed in [MHMP Enterprise].

2. A Host on a multi-homing link would be better compliant with [Default Address] section 5 (source address selection) rule 5 (for different interfaces on the host) and rule 5.5 (for different next-hops on the same interface). It would help to properly choose a source address compliant with the next-hop chosen first.

3. [MHMP Enterprise] proposes a substitution for rule 5.5 absence on hosts by [Conditional PIO] that should not leave a choice to host for what source address to choose.

4. If the source address would be chosen wrongly (because of no support for rule 5.5 of [Default Address] and no support for [Conditional PIO]) then it is still possible to reroute the packet later by source routing proposed in [MHMP Enterprise]. Albeit, the performance would be affected by pushing traffic through redundant routing hop.

The reversal of choice to source address first would permit to improve of the functionality (see next section) and simplify the "equal prefixes" case because it is much easier to choose next-hop after source address by simply excluding default routers that do not advertise particular PIOs.

4. Solution for the case "non-equal prefixes"

This case is more complicated. It is not fully resolved yet in the standardization. It is the primary motivation for the development of this document.

It would be too late to try to solve this problem on a router, because the wrong source address may be already chosen by the host – it would not be possible to contact the appropriate resource in the "walled garden" or filtered for any other purpose. Additionally, the
wrong choice for source address would not permit traffic engineering and host reaction to network quality of services.

Currently, there is only one standardized method to resolve the case of "non-equal prefixes":

1. The same policies could be formatted differently and fed to the host by two mechanisms at the same time: 1) "Routing Information Options" of [Route Preferences] and 2) [Policy by DHCP] to modify policies in [Default Address] selection algorithm. Then the current priority of mechanisms could be preserved the same: initially [ND] or routing would choose the next-hop, then [Default Address] would choose a proper source address. It is the method that is assumed in [MHMP]. This method is complicated and costly, and the probability of acceptance is very low. Moreover, [Policy by DHCP] was not adopted by the market - it is not available on the major operating systems and home gateways.

Alternatively, [Default Address] section 7 discusses the potential capability to reverse the decision’s order: source address may be chosen first, only then to choose next-hop (default router). Then many additional methods are possible for how to choose a source address first:

2. Only policies could be supplied by [Policy by DHCP] to the [Default Address] selection algorithm. This method has a low probability of implementation because of not wide support of DHCPv6 in the industry. Maybe this method would have more acceptance in the future.

3. It is possible to check the longest match between the source and the destination address to choose the potentially closest address. This method looks most promising, it is partially discussed in [Default Address] section 7.

4. The host could use DNS requests with different source addresses to understand what is visible for a particular source address.

5. URL for configuration information could be supplied in RA - see [Provisioning domains].

6. The host may have local performance management capabilities (packet loss, delay, jitter, etc) to choose the best source for the application.

It is possible to have other methods for how the host could decide locally on the best source address as its first decision. This document is readily extensible in this direction.
The source address selected from the proper carrier is the complete information needed for the host to choose the next-hop, but it needs improvement of [ND] and [Default Address]. [ND] default round-robin distribution between available routers should be extended for the host to prioritize default routers that have announced prefixes used for the source address of the considered packet. Section 7 of [Default Address] should be extended and recommended for hosts to support MHMP.

This document’s proposals are inspired by [Multi-Homing] section 3.2. The difference is that the same rules are formulated not as the general advice, but as a particular extension to [ND] and [Default Address] - see section 6. of this document.

5. Resolution for a not reachable provider

Let’s assume the fault situation when one provider is not reachable in the [MHMP] environment. A prefix may be very dynamic for a few reasons. It could be received from some protocols (DHCP-PD, HNCP). The prefix could become invalid (at least for the global Internet connectivity) as a result of the abrupt link loss in the upstream direction to the carrier that distributed this prefix.

Additionally, consider the more complicated case when some hosts on the upstream routed path to this provider may still be reachable using a particular prefix but Internet connectivity is broken later.

Let’s consider the problem. Because Internet connectivity is lost for this prefix, it should be announced to hosts with zero Preferred Lifetime. [Route Preferences] gives the possibility to inform hosts that particular a prefix (RIO) is still available on-site but it would be an automation challenge to dynamically calculate and announce the prefix. Additionally, [Route Preferences] should be supported by hosts. In general, it is not a good idea to involve [ND] in routing. Hence, it is better to support on-site connectivity by ULA that may not be invalidated. There are many reasons to promote [ULA] for internal site connectivity: (1) hosts may not have GUA address at all without initial connection to the provider, (2) PA addresses would be invalidated within 30 days of disconnect anyway, (3) it is not a good idea to use addresses from PA pool that is disconnected from global Internet - hosts may have a better option to get global reachability. ULA has better security (open transport ports that are not accessible from the Internet) which is an additional bonus. It is effectively the request to join current [CPE Requirements] and
[HomeNet Architecture] requirements in sections 2.2, 2.4, 3.4.2 that the subscriber’s network should have local ULA addresses.

Prefix deprecation should be done by RA with zero Lifetime for this prefix. It will put the prefix on hosts to the deprecated status that according to many standards ([ND], [SLAAC], and [Default Address]) would prioritize other addresses. Global communication would be disrupted for this prefix anyway. Local communication for deprecated addresses would continue till normal resolution because the default Valid Lifetime is 30 days. Moreover, if it would happen that this delegated prefix was the only one in the local network (no [ULA] for the same reason), then new sessions would be opened on the deprecated prefix because it is the only address available. If connectivity would be re-established and the same prefix would be delegated to the link - it would be announced again with the proper preferred lifetime. If a different prefix could be delegated by the PA provider, then the old prefix would stay in deprecated status. It is an advantage for the host that would know about global reachability on this prefix (by deprecated status) because the host may use other means for communication at that time.

Such dynamic treatment of prefixes may have the danger of [ND] messages flooding if the link on the path to the PA provider would be oscillating. [HNCP] section 1.1 states: "it is desirable for ISPs to provide large enough valid and preferred lifetimes to avoid unnecessary HNCP state churn in homes". It makes sense to introduce dampening for the rate of prefix announcements.

Such conceptual change in the treatment of prefixes would not affect current enterprise installations where prefixes are static.

It is important to mention again that it is the responsibility of the respective protocol (that has delivered the prefix to the considered router) to inform the router that the prefix is not routed anymore to the respective carrier. It is easy to do it in the simplified topology when the only router could correlate uplink status with the DHCP-PD prefix delegated early. Some additional protocols like [HNCP] are needed for a more complex topology.

There is nothing in [ND] or [SLAAC] that prevents us from treating prefixes as something more dynamic than "renumbering" to reflect the dynamic path status to the PA provider. Section 6.3. proposes extensions to [CPE Requirements] and [SLAAC] that follow the logic of this section.
6. Extensions of the existing standards

The solution is about several standard extensions that are needed to fulfill the solutions discussed above. They are split into separate sections for better understanding.

6.1. Preference to choose source address before the next-hop.

* Section 7 (Interactions with routing) of [Default Address] has at the beginning:

"This specification of source address selection assumes that routing (more precisely, selecting an outgoing interface on a node with multiple interfaces) is done before source address selection. However, implementations MAY use source address considerations as a tiebreaker when choosing among otherwise equivalent routes."

Replace the above text with the text:

"This specification of source address selection did assume that routing (more precisely, selecting an outgoing interface on a node with multiple interfaces) is done after source address selection. MHMP support strongly demands choosing the source address first. Hence, an implementation SHOULD change the preference to source address choice first. There are a few methods below for how to choose a source address for any particular destination. The list is not exhaustive - it should be augmented later. The implementation MAY develop their method for choosing source address first."

The next 2 paragraphs of the original RFC 6724 should be preserved. The one is about choosing the source address that has the longest much with the destination address. Another one is equivalent to the methods proposed in [Conditional PIO].

Add 4 new methods for source address choice at the end of the section:

"The [Default Address] policy table may be updated by [Policy by DHCP] to guide source address selection.

The implementation may generate DNS requests from an address of every IPv6 PIO available to make sure that a particular source address has the reachability to the resource (split DNS may be implemented for "walled garden")."
URL for configuration information could be supplied in RA - see [Provisioning domains].

The host may have local performance management capabilities (packet loss, delay, jitter, etc) to choose the best source for the application."

6.2. Default router choice by host

* Section 6.3.6 (Default Router Selection) of [ND], add an initial policy to default router selection:

0) For the cases when a particular implementation of ND does know the source address at the time of default router selection (it means that the source address was chosen first), then default routers that advertise the prefix for the respective source address SHOULD be preferred over routers that do not advertise the respective prefix.

6.3. Prefixes become dynamic

* This document joins the request to [CPE Requirements] that has been proposed in section 11 (General Requirements for HNCP Nodes) of [HNCP]:

The requirement L-13 to deprecate prefixes is applied to all delegated prefixes in the network from which assignments have been made on the respective interface. Furthermore, the Prefix Information Options indicating deprecation MUST be included in Router Advertisements for the remainder of the prefixes’ respective valid lifetime, but MAY be omitted after at least 2 hours have passed.

* Add section 4.2 into [SLAAC]:

4.2 Dynamic Link Renumbering

Prefix delegation (primarily by DHCP-PD) is adopted by the industry as the primary mechanism of PA address delegation in the fixed and mobile broadband environments, including cases of small businesses and branches of the big enterprises. The delegated prefix is tied to a dynamic link that has a considerable probability to be disconnected, especially in a mobile environment. The delegated prefix is losing the value if the remote
site is disconnected from the prefix provider - this fact should be propagated to all nodes on the disconnected site, including hosts. Information Options indicating deprecation (multicast RA with zero Preferred Lifetime) MUST be sent at least one time. It SHOULD be included in Router Advertisements for the remainder of the prefixes' respective valid lifetime but MAY be omitted after 2 hours of deprecation announcements.

There is a high probability that connectivity to the provider would be restored very soon then the prefix could be announced again to all nodes on the site.

There is the probability that in a small period the same problem would disconnect the site again (especially for mobile uplink). Such oscillation between available and not available providers could happen frequently that would flood the remote site with [ND] updates.

A dampening mechanism MAY be implemented to suppress oscillation:
if the time between a particular prefix announcement and previous deprecation was less than DampeningCheck then delay the next prefix announcement for DampeningDelay and check the need for the prefix announcement after DampeningDelay seconds.

It is recommended for protocol designers to implement a dampening mechanism for protocols (like [HNCP]) that would be used to distribute prefix delegation inside the site to relieve the majority of site routers and the protocol itself from the processing of oscillating messages.

* Section 5.1 (Node Configuration Variables) of [SLAAC], add timers:

DampeningCheck - the time between prefix announcement and previous deprecation is checked against this value to decide about the dampening need. The timer should use a 16bit unsigned integer measured in seconds. The default value is 10 seconds.

DampeningDelay - the delay (penalty) for the next attempt to announce the same prefix again. The timer should use a 16bit unsigned integer measured in seconds. The default value is 10 seconds.

These timers should be configurable like all other timers in [SLAAC] section 5.1.
6.4. Default router announcement rules

* This document joins [HNCP] section 11 (General Requirements for HNCP Nodes) request to [CPE Requirements]:

The generic requirements G-4 and G-5 are relaxed such that any known default router on any interface is sufficient for a router to announce itself as the default router; similarly, only the loss of all such default routers results in self-invalidation.

6.5. Clean orphaned prefixes after default router list change

* Section 6.3.6 (Timing out Prefixes and Default Routers) of [ND] has:

"Whenever the Lifetime of an entry in the Default Router List expires, that entry is discarded. When removing a router from the Default Router list, the node MUST update the Destination Cache in such a way that all entries using the router perform next-hop determination again rather than continue sending traffic to the (deleted) router."

Add at the end:

"All prefixes announced by deprecated default router SHOULD be checked on the announcement from other default routers. If any prefix is not anymore announced from any router - it SHOULD be deprecated."

7. Interoperability analysis

This document mostly intersects with Homenet working group documents [HomeNet Architecture], [HNCP], and [MHMP]. This document simplifies the discussed in the [MHMP] solution of updating 2 tables on the host (routing and default address selection policy) by reversing the choice for the source address first.

[CPE Requirements] have the assumption of managing simplified topologies by manipulating routing information injection into [ND]. It has been shown in [MHMP] and in this document that it is better to signal reachability information to the host by the deprecation of delegated prefixes. This document joins [MHMP] request to change the approach.

Expires January 1, 2023
This document does not contradict in any way to [Conditional PIO] or [MHMP Enterprise] that explain in detail the "equal prefixes" case but expend MHMP solution to the "non-equal prefixes" case.

[Happy Eyeballs] are sorting destination addresses. The proposals of this document are coming into the discussion after the destination addresses are chosen. Hence, [Happy Eyeballs] operation is not impacted.

[Route Preferences] have been avoided as the mechanism for environments with PA address space because it is better to select the source address first for the more general case. [Route Preferences] could still be applicable for PI (Provider-Independent) address environments where only next-hops need to be chosen properly.

8. Security Considerations

This document does not introduce new vulnerabilities.

9. IANA Considerations

This document has no request to IANA.

10. References

10.1. Normative References


10.2. Informative References


Expires January 1, 2023


11. Acknowledgments

Thanks to the 6man working group for problem discussion.

Authors’ Addresses

Eduard Vasilenko
Huawei Technologies
17/4 Krylatskaya st, Moscow, Russia 121614
Email: vasilenko.eduard@huawei.com

Paolo Volpato
Huawei Technologies
Via Lorenteggio 240, 20147 Milan, Italy
Email: paolo.volpato@huawei.com
ICMPv6 Echo Request/Reply for Enabled In-situ OAM Capabilities
draft-xiao-6man-icmpv6-ioam-conf-state-01

Abstract

This document describes the ICMPv6 IOAM Echo functionality, which uses the ICMPv6 IOAM Echo Request/Reply messages, allowing the IOAM encapsulating node to discover the enabled IOAM capabilities of each IOAM transit and decapsulating node.

This document updates RFC 4884.

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 27 October 2022.

Copyright Notice

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

IPv6 encapsulation for In-situ OAM (IOAM) data is defined in [I-D.ietf-ippm-ioam-ipv6-options], which uses IPv6 hop-by-hop options and destination option to carry IOAM data.

As specified in [I-D.ietf-ippm-ioam-conf-state], echo request/reply can be used for the IOAM encapsulating node to discover the enabled IOAM capabilities at IOAM transit and decapsulating nodes.
As specified in [RFC4443], the Internet Control Message Protocol for IPv6 (ICMPv6) is an integral part of IPv6, and the base protocol MUST be fully implemented by every IPv6 node. ICMPv6 messages include error messages and informational messages, and the latter are referred to as ICMPv6 Echo Request/Reply messages. [RFC4884] defines ICMPv6 Extension Structure by which multi-part ICMPv6 error messages are supported. [RFC8335] defines ICMPv6 Extended Echo Request/Reply messages, and the ICMPv6 Extended Echo Request contains an ICMPv6 Extension Structure customized for this message. Both [RFC4884] and [RFC8335] provide sound principles and examples on how to extend ICMPv6 error messages and echo request/reply messages.

This document describes the ICMPv6 IOAM Echo functionality, which uses the ICMPv6 IOAM Echo Request/Reply messages, allowing the IOAM encapsulating node to discover the enabled IOAM capabilities of each IOAM transit and decapsulating node.

The IOAM encapsulating node sends an ICMPv6 IOAM Echo Request message to each IOAM transit and decapsulating node, then each receiving node executes access control procedures, and if access is granted, each receiving node returns an ICMPv6 IOAM Echo Reply message which indicates the enabled IOAM capabilities of the receiving node. The ICMPv6 IOAM Echo Reply message contains an ICMPv6 Extension Structure exactly customized to this message, and the ICMPv6 Extension Structure contains one or more IOAM Capabilities Objects.

Note that before the IOAM encapsulating node sends the ICMPv6 IOAM Echo Request messages, it needs to know the IPv6 address of each node along the transport path of a data packet to which IOAM data would be added. That can be achieved by executing ICMPv6 traceroute or provisioning explicit path at the IOAM encapsulating node.

2. Conventions Used in This Document

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. ICMPv6 IOAM Echo Request

The ICMPv6 IOAM Echo Request message is encapsulated in an IPv6 header [RFC8200], like any ICMPv6 message.

The ICMPv6 IOAM Echo Request message has the following format:
IPv6 Header fields:
* Source Address: The Source Address identifies the IOAM encapsulating node. It MUST be a valid IPv6 unicast address.
* Destination Address: The Destination Address identifies the IOAM transit or decapsulating node. It MUST be a valid IPv6 unicast address.

ICMPv6 fields:
* Type: IOAM Echo Request. The value is TBD1.
* Code: MUST be set to 0 and MUST be ignored upon receipt.
* Checksum: The same as defined in [RFC4443].
* Identifier: An Identifier aids in matching IOAM Echo Replies to IOAM Echo Requests. It may be zeroed.
* Sequence Number: A Sequence Number to aid in matching IOAM Echo Replies to IOAM Echo Requests. It may be zeroed.
* Num of NS-IDs: Number of Namespace-IDs within the payload.
* Following the IOAM Echo Request header, it's a List of Namespace-IDs, which is also called IOAM Capabilities Query Container Payload in Section 3.1 of [I-D.ietf-ippm-ioam-conf-state]. If the payload would not otherwise terminate on a 4-octet boundary, it MUST be padded with zeroes.
4. ICMPv6 IOAM Echo Reply

The ICMPv6 IOAM Echo Reply message is encapsulated in an IPv6 header [RFC8200], like any ICMPv6 message.

The ICMPv6 IOAM Echo Reply message has the following format:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>Checksum</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Sequence Number</td>
<td>Num of NS-IDs</td>
<td></td>
</tr>
</tbody>
</table>

.IOAM Capabilities Response Container Payload.

as specified in .

Section 3.2 of draft-ietf-ippm-ioam-conf-state .

Figure 2: ICMPv6 IOAM Echo Reply Message

IPv6 Header fields:

* Source Address: Copied from the Destination Address field of the invoking IOAM Echo Request packet.

* Destination Address: Copied from the Source Address field of the invoking IOAM Echo Request packet.

ICMPv6 fields:

* Type: IOAM Echo Reply. The value is TBD2.

* Code: Values are (0) No Error, (1) Malformed Query, (2) No Matched Namespace-ID, and (3) Exceed the minimum IPv6 MTU.

* Checksum: The same as defined in [RFC4443].

* Identifier: Copied from the Identifier field of the invoking IOAM Echo Request message.

* Sequence Number: Copied from the Sequence Number field of the invoking IOAM Echo Request message.

* Num of NS-IDs: Number of different Namespace-IDs within the payload, its value MUST be no more than the Num of NS-IDs field of the invoking IOAM Echo Request message.
Following the IOAM Echo Reply header, it’s a List of IOAM Capabilities Objects, which is also called IOAM Capabilities Response Container Payload in Section 3.2 of [I-D.ietf-ippm-ioam-conf-state].

Section 7 of [RFC4884] defines the ICMP Extension Structure. As per RFC 4884, the Extension Structure contains exactly one Extension Header followed by one or more objects. When applied to the ICMPv6 IOAM Echo Reply message, the ICMP Extension Structure MUST contain one or more IOAM Capabilities Objects.

4.1. IOAM Capabilities Objects

All ICMPv6 IOAM Capabilities Objects are encapsulated in an ICMPv6 IOAM Echo Reply message.

Each ICMPv6 IOAM Capabilities Object has the following format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Length            |   Class-Num   |   C-Type      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                IOAM Capabilities Object Payload               .
.                        as specified in                        .
.        Section 3.2.x of draft-ietf-ippm-ioam-conf-state       .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: IOAM Capabilities Object

Object fields:

* Class-Num: IOAM Capabilities Objects. The values are listed as the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Object Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD3</td>
<td>IOAM Tracing Capabilities Object</td>
</tr>
<tr>
<td>TBD4</td>
<td>IOAM Proof-of-Transit Capabilities Object</td>
</tr>
<tr>
<td>TBD5</td>
<td>IOAM Edge-to-Edge Capabilities Object</td>
</tr>
<tr>
<td>TBD6</td>
<td>IOAM DEX Capabilities Object</td>
</tr>
<tr>
<td>TBD7</td>
<td>IOAM End-of-Domain Object</td>
</tr>
</tbody>
</table>

* C-Type: Values are listed as the following:
<table>
<thead>
<tr>
<th>Class-Num</th>
<th>C-Type</th>
<th>C-Type Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD3</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>Pre-allocated Tracing</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Incremental Tracing</td>
<td></td>
</tr>
<tr>
<td>TBD4</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>TBD5</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>TBD6</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>TBD7</td>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

* Length: Length of the object, measured in octets, including the Object Header and Object Payload.

* Following the IOAM Capabilities Object Header, it’s the IOAM Capabilities Object Payload, which is defined respectively in Section 3.2.1, Section 3.2.2, Section 3.2.3, Section 3.2.4, Section 3.2.5 and Section 3.2.6 of [I-D.ietf-ippm-ioam-conf-state].

4.2. Examples of IOAM Echo Reply

The format of ICMPv6 IOAM Echo Reply can vary from deployment to deployment.

In a deployment where only the default Namespace-ID is used, the IOAM Pre-allocated Tracing Capabilities and IOAM Proof-of-Transit Capabilities are enabled at the IOAM transit node that received ICMPv6 IOAM Echo Request message, the ICMPv6 IOAM Echo Reply message is depicted as the following:
Figure 4: Example 1 of IOAM Echo Reply

In a deployment where two Namespace-IDs (Namespace-ID1 and Namespace-ID2) are used, for both Namespace-ID1 and Namespace-ID2 the IOAM Pre-allocated Tracing Capabilities and IOAM Proof-of-Transit Capabilities are enabled at the IOAM transit node that received ICMPv6 IOAM Echo Request message, the ICMPv6 IOAM Echo Reply message is depicted as the following:
In a deployment where only the default Namespace-ID is used, the IOAM Pre-allocated Tracing Capabilities, IOAM Proof-of-Transit Capabilities and IOAM Edge-to-Edge Capabilities are enabled at the IOAM decapsulating node that received ICMPv6 IOAM Echo Request message, the ICMPv6 IOAM Echo Reply message is depicted as the following:

```
<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Sequence Number</td>
<td>Num of NS-IDs</td>
</tr>
<tr>
<td>Length</td>
<td>Class-Num</td>
<td>C-Type</td>
</tr>
<tr>
<td>IOAM-Trace-Type</td>
<td>Reserved</td>
<td>W</td>
</tr>
<tr>
<td>Namespace-ID1</td>
<td>Ingress_MTU</td>
<td></td>
</tr>
<tr>
<td>Ingress_if_id (short or wide format)</td>
<td>......</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>Class-Num</td>
<td>C-Type</td>
</tr>
<tr>
<td>Namespace-ID1</td>
<td>IOAM-POT-Type</td>
<td>SoP</td>
</tr>
<tr>
<td>Length</td>
<td>Class-Num</td>
<td>C-Type</td>
</tr>
<tr>
<td>IOAM-Trace-Type</td>
<td>Reserved</td>
<td>W</td>
</tr>
<tr>
<td>Namespace-ID2</td>
<td>Ingress_MTU</td>
<td></td>
</tr>
<tr>
<td>Ingress_if_id (short or wide format)</td>
<td>......</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>Class-Num</td>
<td>C-Type</td>
</tr>
<tr>
<td>Namespace-ID2</td>
<td>IOAM-POT-Type</td>
<td>SoP</td>
</tr>
</tbody>
</table>
```

Figure 5: Example 2 of IOAM Echo Reply
Figure 6: Example 3 of IOAM Echo Reply

Note that when an ICMPv6 IOAM Echo Request message or IOAM Echo Reply message is received, the Payload Length field of IPv6 Header [RFC8200] indicates the message length.

5. ICMPv6 Message Processing

When a node receives an ICMPv6 IOAM Echo Request and any of the following conditions apply, the node MUST silently discard the incoming message:

* The node does not recognize the ICMPv6 IOAM Echo Request message.

* The node has not explicitly enabled ICMPv6 IOAM Echo functionality.

* The incoming ICMPv6 IOAM Echo Request carries a Source Address that is not explicitly authorized.

* The Source Address of the incoming message is not a unicast address.
The Destination Address of the incoming message is a multicast address.

Otherwise, when a node receives an ICMPv6 IOAM Echo Request, it MUST format an ICMPv6 IOAM Echo Reply as follows:

- Set the Hop Limit to 255.
- Set the DiffServ codepoint to CS0 [RFC4594].
- Copy the Destination Address from the IOAM Echo Request to the Source Address of the IOAM Echo Reply.
- Copy the Source Address from the IOAM Echo Request to the Destination Address of the IOAM Echo Reply.
- Set the Next Header to (58) ICMPv6.
- Set the ICMPv6 Type to (TBD2) IOAM Echo Reply.
- Copy the Identifier from the IOAM Echo Request to the IOAM Echo Reply.
- Copy the Sequence Number from the IOAM Echo Request to the IOAM Echo Reply.
- Set the Code field as described in Section 5.1.
- If the Code field is equal to (0) No Error, then add one or more objects as described in Section 4.1.
- Set the Checksum appropriately.
- Forward the ICMPv6 IOAM Echo Reply to its destination.

### 5.1. Code Field Processing

The Code field MUST be set to (1) Malformed Query if any of the following conditions apply:

- The ICMPv6 IOAM Echo Request does not include any Namespace-ID.
- The value of Num of NS-IDs field does not match the contained list of Namespace-IDs.
- The query is otherwise malformed.
The Code field MUST be set to (2) No Matched Namespace-ID if none of the contained list of Namespace-IDs is recognized.

The Code field MUST be set to (3) Exceed the minimum IPv6 MTU if the formatted ICMPv6 IOAM Echo Reply exceeds the minimum IPv6 MTU (i.e., 1280 octets). In this case, all objects MUST be stripped before forwarding the ICMPv6 Echo Reply to its destination.

Otherwise, the Code field MUST be set to (0) No Error.

6. Updates to RFC 4884

Section 4.6 of [RFC4884] provides a list of extensible ICMP messages (i.e., messages that can carry the ICMP Extension Structure). This document adds the ICMPv6 IOAM Echo Request message and the ICMPv6 IOAM Echo Reply message to that list.

7. IANA Considerations

This document requests the following IANA actions:

* Add the following to the "ICMPv6 'type' Numbers" registry:
  - TBD1 IOAM Echo Request
    - As ICMPv6 distinguishes between informational and error messages, and this is an informational message, the value must be assigned from the range 128-255.

* Add the following to the "Type TBD1 - IOAM Echo Request" sub-registry:
  - (0) No Error

* Add the following to the "ICMPv6 'type' Numbers" registry:
  - TBD2 IOAM Echo Reply
    - As ICMPv6 distinguishes between informational and error messages, and this is an informational message, the value must be assigned from the range 128-255.

* Add the following to the "Type TBD2 - IOAM Echo Reply" sub-registry:
  - (0) No Error
  - (1) Malformed Query
- (2) No Matched Namespace-ID
- (3) Exceed the minimum IPv6 MTU

* Add the following to the "ICMP Extension Object Classes and Class Sub-types" registry:
  - (TBD3) IOAM Tracing Capabilities Object

* Add the following C-types to the "Sub-types - Class TBD3 - IOAM Tracing Capabilities Object" sub-registry:
  - (0) Reserved
  - (1) Pre-allocated Tracing
  - (2) Incremental Tracing
  - C-Type values are assigned on a First Come First Serve (FCFS) basis with a range of 0-255.

* Add the following to the "ICMP Extension Object Classes and Class Sub-types" registry:
  - (TBD4) IOAM Proof-of-Transit Capabilities Object

* Add the following C-types to the "Sub-types - Class TBD4 - IOAM Proof-of-Transit Capabilities Object" sub-registry:
  - (0) Reserved
  - C-Type values are assigned on an FCFS basis with a range of 0-255.

* Add the following to the "ICMP Extension Object Classes and Class Sub-types" registry:
  - (TBD5) IOAM Edge-to-Edge Capabilities Object

* Add the following C-types to the "Sub-types - Class TBD5 - IOAM Edge-to-Edge Capabilities Object" sub-registry:
  - (0) Reserved
  - C-Type values are assigned on an FCFS basis with a range of 0-255.
* Add the following to the "ICMP Extension Object Classes and Class Sub-types" registry:

  - (TBD6) IOAM DEX Capabilities Object

* Add the following C-types to the "Sub-types - Class TBD6 - IOAM DEX Capabilities Object" sub-registry:

  - (0) Reserved
  - C-Type values are assigned on an FCFS basis with a range of 0-255.

* Add the following to the "ICMP Extension Object Classes and Class Sub-types" registry:

  - (TBD7) IOAM End-of-Domain Object

* Add the following C-types to the "Sub-types - Class TBD7 - IOAM End-of-Domain Object" sub-registry:

  - (0) Reserved
  - C-Type values are assigned on an FCFS basis with a range of 0-255.

All codes mentioned above are assigned on an FCFS basis with a range of 0-255.

8. Security Considerations

Security issues discussed in [I-D.ietf-ippm-ioam-conf-state] apply to this document.

This document recommends using IP Authentication Header [RFC4302] or IP Encapsulating Security Payload Header [RFC4303] to provide integrity protection for IOAM Capabilities information.

This document recommends using IP Encapsulating Security Payload Header [RFC4303] to provide privacy protection for IOAM Capabilities information.

This document recommends that the network operators establish policies that restrict access to ICMPv6 IOAM Echo functionality. In order to enforce these policies, nodes that support ICMPv6 IOAM Echo functionality MUST support the following configuration options:
Enable/disable ICMPv6 IOAM Echo functionality. By default, ICMPv6 IOAM Echo functionality is disabled.

Define enabled Namespace-IDs. By default, all Namespace-IDs except the default one (i.e., Namespace-ID 0x0000) are disabled.

For each enabled Namespace-ID, define the prefixes from which ICMPv6 IOAM Echo Request messages are permitted.

When a node receives an ICMPv6 IOAM Echo Request message that it is not configured to support, it MUST silently discard the message. See Section 5 for details.

In order to protect local resources, implementations SHOULD rate-limit incoming ICMPv6 IOAM Echo Request messages.

9. Acknowledgements

TBA.

10. References

10.1. Normative References

[I-D.ietf-ippm-ioam-conf-state]


Min & Mirsky         Expires 27 October 2022
10.2. Informative References

[I-D.ietf-ippm-ioam-ipv6-options]
Bhandari, S. and F. Brockners, "In-situ OAM IPv6 Options",
Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-ipv6-options-07, 6 February 2022,

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302,
DOI 10.17487/RFC4302, December 2005,

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)",
RFC 4303, DOI 10.17487/RFC4303, December 2005,

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration
Guidelines for DiffServ Service Classes", RFC 4594,
DOI 10.17487/RFC4594, August 2006,

(IPv6) Specification", STD 86, RFC 8200,
DOI 10.17487/RFC8200, July 2017,

[RFC8335] Bonica, R., Thomas, R., Linkova, J., Lenart, C., and M.
Boucadair, "PROBE: A Utility for Probing Interfaces",
RFC 8335, DOI 10.17487/RFC8335, February 2018,

Authors’ Addresses

Xiao Min
ZTE Corp.
Nanjing
China
Phone: +86 25 88013062
Email: xiao.min2@zte.com.cn
Greg Mirsky
Ericsson
United States of America
Email: gregimirsky@gmail.com
Abstract

This document provides a unified mechanism that makes the upper-layer checksum computation rule defined in IPv6 Specification applicable, whether SRv6 SIDs or SRv6 compressed SIDs are used.

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 19 November 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) 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.
1.  Introduction

IPv6 Specification [RFC8200] defines how upper-layer checksum is computed. Specifically, a "pseudo-header" for IPv6 is constructed as a portion of fields included in upper-layer (e.g., TCP, UDP, ICMPv6, OSPF) checksum computation. As defined in Section 8.1 of [RFC8200], if the IPv6 packet doesn’t contain Routing header, the Destination Address used in the pseudo-header will be in the Destination Address field of the IPv6 header; if the IPv6 packet contains a Routing header, the Destination Address used in the pseudo-header is that of the final destination. In the latter case, at the originating node, that address will be in the last element of the Routing header; at the recipient(s), that address will be in the Destination Address field of the IPv6 header. As also defined in Section 8.1 of [RFC8200], any node implementing zero-checksum mode of UDP tunnel must follow the requirements specified in "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums" [RFC6936], and it’s outside the scope of this document.

Segment Routing over IPv6 (SRv6) [RFC8754] defines an IPv6 Routing header called Segment Routing Header (SRH). To comply with the upper-layer checksum computation rule defined in [RFC8200], at the SRv6 ingress node, the last element of the SRH, i.e., the last Segment Identifier (SID), will become the Destination Address used in the pseudo-header for upper-layer checksum computation; at the SRv6 egress node, after SRH processing is finished, the Destination Address in the IPv6 header will become the Destination Address used in the pseudo-header for upper-layer checksum computation. Note that even at the SRv6 egress node, SRH processing may still invoke IPv6 Destination Address substitution.
The C-SID document [I-D.ietf-spring-srv6-srh-compression] defines SRv6 compressed SIDs, which use 16-bit or 32-bit SRv6 C-SID to substitute 128-bit SRv6 SID. The NEXT-C-SID flavor and REPLACE-C-SID flavor are defined in the C-SID document. In one case of NEXT-C-SID flavor, at the SRv6 ingress node, the IPv6 packet doesn’t contain Routing header, more than one C-SIDs are included in IPv6 Destination Address, the upper-layer checksum computation rule defined in [RFC8200] doesn’t apply anymore. In another case of REPLACE-C-SID flavor, at the SRv6 ingress node, the IPv6 packet contains an SRH, the last element of the SRH is not a 128-bit IPv6 address, but a 16-bit or 32-bit C-SID, the upper-layer checksum computation rule defined in [RFC8200] doesn’t apply anymore.

This document provides a unified mechanism that makes the upper-layer checksum computation rule defined in IPv6 Specification applicable, whether SRv6 SIDs or SRv6 compressed SIDs are used.

2. Conventions

2.1. Requirements Language

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

2.2. Abbreviations

SR: Segment Routing
SRv6: Segment Routing with IPv6 data plane
SID: Segment ID
C-SID: Compressed Segment ID [I-D.ietf-spring-srv6-srh-compression]
SRH: Segment Routing Header [RFC8754]
PSP: Penultimate Segment Pop of the SRH [RFC8986]
TCP: Transmission Control Protocol [RFC0793]
UDP: User Datagram Protocol [RFC0768]
ICMPv6: Internet Control Message Protocol for IPv6 [RFC4443]
OSPF: Open Shortest Path First protocol [RFC2328]
3. Unified Mechanism for Upper-Layer Checksum in SRv6

This section defines a unified mechanism for upper-layer checksum in SRv6 networks. This mechanism utilizes a new SRH flag and requests all SRv6 nodes along the transport path to act on the new SRH flag.

3.1. C-flag in Segment Routing Header

[RFC8754] describes the Segment Routing Header (SRH) and how SRv6 capable nodes use it. The SRH contains an 8-bit "Flags" field. This document defines the following bit in the SRH Flags field to carry the C-flag:

```
0 1 2 3 4 5 6 7
+++-+-+-+-+-+-+-+
|     |C|       |
+++-+-+-+-+-+-+-+
```

Where:

C-flag: Checksum flag in the SRH Flags field defined in [RFC8754]. When C-flag is set, the last element of the SRH MUST be set to an IPv6 address of the final destination.

3.2. C-flag Processing

The C-flag in SRH is used as a marking-bit in the SRv6 packets using upper-layer checksum, each segment endpoint would process the C-flag as defined in this document, to make the SRv6 upper-layer checksum computation smooth and complied to [RFC8200].

At the upper-layer checksum originating node, if the IPv6 packet contains an SRH, the SRH C-flag MUST be set and the Segment List[0] MUST be set to a 128-bit IPv6 address of the final destination; if the IPv6 packet doesn’t contain an SRH while the Destination Address field contains more than one compressed SID, an SRH MUST be added with C-flag set and Segment List[0] set to a 128-bit IPv6 address of the final destination. When the upper-layer checksum originating node knows more than one IPv6 address of the final destination, e.g., a local interface address of the final destination, a 128-bit SID locally instantiated at the final destination, and an IPv6 address transformed from a 16-bit or 32-bit compressed SID locally instantiated at the final destination, the originating node needs to select one of them as the last element of SRH, how the originating node makes the choice is beyond the scope of this document.
When the penultimate segment of a segment-list is a Penultimate Segment Pop (PSP) SID, the SRH is removed at the penultimate segment and the C-flag is not processed at the ultimate segment. The penultimate segment as a PSP SID MUST copy Segment List[0] from the SRH to the Destination Address field of the IPv6 header, then the ultimate segment can still compute the upper-layer checksum with correct IPv6 Destination Address even without SRH.

When an SRv6 node receives a packet destined to S and S is a local SID, the line S01 of the pseudo-code associated with the SID S, as defined in Section 4.3.1.1 of [RFC8754], is appended to as follows for the C-flag processing.

S01.2. IF C-flag is set and local configuration permits C-flag processing {
    If (Segment List[0] is locally instantiated or represents a local interface) {
        a. Set Segments Left to 0.
        b. Update IPv6 DA with Segment List[0].
    }
    Else {
        If (IPv6 DA is locally instantiated as a PSP SID) {
            a. Update IPv6 DA with Segment List[0].
            b. Submit the packet to the egress IPv6 FIB lookup for transmission to the new destination.
        }
    }
}

Note that the C-flag processing happens before execution of regular processing of the local SID S. Specifically, the line S01.2 of the pseudo-code specified in this document is inserted between line S01 and S02 of the pseudo-code defined in Section 4.3.1.1 of [RFC8754]. When the C-flag defined in this document and the O-flag defined in Section 2.1 of [I-D.ietf-6man-spring-srv6-oam] are both set, the C-flag processing happens after O-flag processing. Specifically, the line S01.2 of the pseudo-code specified in this document is inserted between line S01.1 of the pseudo-code defined in Section 2.1.1 of [I-D.ietf-6man-spring-srv6-oam] and line S02 of the pseudo-code defined in Section 4.3.1.1 of [RFC8754].

Also note that if the final destination needs to be reached more than once on the programmed transport path, the SRv6 packets with C-flag set would be terminated at the first time the final destination is reached. If it’s deemed necessary for the SRv6 packets with C-flag set to reach the final destination more than once, more judgment conditions may be added to the pseudo-code of C-flag processing.
4. IANA Considerations

In the "Segment Routing Header Flags" registry created for [RFC8754], a new Checksum Flag is requested from IANA as follows:

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Symbol</th>
<th>Description</th>
<th>Semantics Definition</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>C</td>
<td>Checksum Flag</td>
<td>Section 3.1</td>
<td>This Document</td>
</tr>
</tbody>
</table>

Table 1: New SRH Flag

5. Security Considerations

This document does not raise additional security issues beyond those of the specifications referred to in the list of references.

6. Acknowledgements

TBA.

7. References

7.1. Normative References

[I-D.ietf-6man-spring-srv6-oam]


7.2. Informative References

[I-D.ietf-spring-srv6-srh-compression]


Authors’ Addresses

Xiao Min
ZTE Corp.
Nanjing
China
Phone: +86 25 88013062
Email: xiao.min2@zte.com.cn

Yao Liu
ZTE Corp.
Nanjing
China
Email: liu.yao71@zte.com.cn

Chongfeng Xie
China Telecom
Beijing
China
Email: xiechf@chinatelecom.cn