This document defines a new IPv6 Routing header type, called the Compressed Routing Header (CRH). SRv6+ nodes use the CRH to steer packets from segment to segment along SRv6+ paths.
1. Introduction

This document defines a new IPv6 [RFC8200] Routing header type, called the Compressed Routing Header (CRH). SRv6+ [I-D.bonica-spring-srv6-plus] nodes use the CRH to steer packets from segment to segment along SRv6+ paths.
For details regarding SRv6+ paths, segments, Segment Identifiers (SIDs) and instructions, see [I-D.bonica-spring-srv6-plus].

2. Requirements Language

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

3. The Compressed Routing Header (CRH)

Figure 1: Compressed Routing Header (CRH)

Figure 1 depicts the CRH. The CRH contains the following fields:

- Next Header - Defined in [RFC8200].
- Hdr Ext Len - Defined in [RFC8200].
- Routing Type - Defined in [RFC8200]. Value TBD by IANA. (Suggested value: 5)
- Segments Left - Defined in [RFC8200].
- Last Entry - 8 bits. Represents the zero-based index of the last element of the Segment List.
- Com (Compression) - 2 bits. Represents the length of each entry in the SID List. Values are reserved (0), sixteen bits (1), thirty-two bits (2), and reserved (3). In order to maximize header compression, this value should reflect the smallest feasible Maximum SID Value (MSV). See Section 5.1 of [I-D.bonica-spring-srv6-plus] for MSV details.
- Reserved - SHOULD be set to zero by the sender. MUST be ignored by the receiver.

- SID List - Represents the SRv6+ path as an ordered list of SIDs. SIDs are listed in reverse order, with SID[0] representing the final segment, SID[1] representing the penultimate segment, and so forth. SIDs are listed in reverse order so that Segments Left can be used as an index to the SID List. The SID indexed by Segments Left is called the current SID.

Figure 2 and Figure 3 illustrate CRH encodings with Com equal to 1 and 2. In all cases, the CRH MUST end on a 64-bit boundary. Therefore, the CRH MAY be padded with zeros.

```
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  | Routing Type  | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Last Entry  |Com|              Reserved                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             SID[0]            |          SID[1]               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
//                                                              //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             SID[n]                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Figure 2: Sixteen-bit Encoding (Com equals 1)**

```
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  | Routing Type  | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Last Entry  |Com|              Reserved                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             SID[0]                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             SID[1]                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                                                              //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             SID[n]                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Figure 3: Thirty-two bit Encoding (Com equals 2)**
4. Segment Forwarding Information Base (SFIB)

A segment ingress node MUST maintain one Segment Forwarding Information Base (SFIB) entry for each segment that it originates. Each SFIB entry contains the following information:

- A SID
- A segment type
- Topological instruction parameters

The following are valid segment types:

- Strictly-routed
- Loosely-routed

The following parameters are associated with topological instructions that control strictly-routed segments:

- An IPv6 address that identifies an interface on the segment egress node.
- A primary interface identifier.
- Zero or more secondary interface identifiers.

Loosely-routed segments are associated with a single topological instruction parameter. This parameter is an IPv6 address that identifies an interface on the segment egress node.

5. Processing Rules

5.1. General

[RFC8200] defines rules that apply to IPv6 extension headers, in general, and IPv6 Routing headers, in particular. All of these rules apply to the CRH.

For example:

- 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.
If, while processing a received packet, a node encounters a Routing header with an unrecognized Routing Type value, the required behavior of the node depends on the value of the Segments Left field. If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing header. If Segments Left is non-zero, the node must discard the packet and send an ICMPv6 [RFC4443] Parameter Problem, Code 0, message to the packet’s Source Address, pointing to the unrecognized Routing Type.

If, after processing a Routing header of a received packet, an intermediate node determines that the packet is to be forwarded onto a link whose link MTU is less than the size of the packet, the node must discard the packet and send an ICMPv6 Packet Too Big message to the packet’s Source Address.

5.2. CRH Specific

When a node recognizes and processes a CRH, it executes the following procedure:

- If the IPv6 Source Address is a link-local address, discard the packet.
- If the IPv6 Source Address is a multicast address, discard the packet.
- If Segments Left equal 0, skip over the CRH and process the next header in the packet.
- If Segments Left is greater than Last Entry plus one, discard the packet and send an ICMPv6 Parameter Problem, Code 0, message to the Source Address, pointing to the Segments Left field.
- If Com is equal to (0) or (3) Reserved, discard the packet and send an ICMPv6 Parameter Problem, Code 0, message to the Source Address, pointing to the Com field.
- Compute L, the minimum CRH length (See Section 5.2.1)
- If L is equal to zero or L is greater than Hdr Ext Len, discard the packet and send an ICMPv6 Parameter Problem, Code 0, message to the Source Address, pointing to the Last Entry field.
- Decrement the packet’s Hop Count.
o If the Hop Count has expired, discard the packet and send an ICMPv6 Time Expired message to the packet’s source node.

o Decrement Segments Left

o Search for the current SID in the SFIB.

o If the above-mentioned search does not return an SFIB entry, discard the packet and send an ICMPv6 Parameter Problem, Code 0, message to the Source Address, pointing to the current SID.

o If the above-mentioned search returns an SFIB entry and the segment type is strictly-routed, execute the strictly-routed topological instruction described in Section 5.2.2.

o If the above-mentioned search returns an SFIB entry and the segment type is loosely-routed, execute the loosely-routed topological instruction described in Section 5.2.3.

The above stated rules are demonstrated in Appendix A.

5.2.1. Computing Minimum CRH Length

The algorithm described in this section accepts the following CRH fields as its input parameters:

o Compression (Com).

o Last Entry.

It yields L, the minimum CRH length. The minimum CRH length is measured in 8-octet units, not including the first 8 octets.
<CODE BEGINS>

if (Com == 1) { /* Sixteen bit encoding */
    L = ( ( Last Entry + 1 ) / 4 );
    if ( ( Last Entry + 1 ) % 4 )
        L++;
}
elsif (Com == 2) { /* Thirty-two bit encoding */
    L = ( ( Last Entry + 1 ) / 2 );
    if ( ( Last Entry + 1 ) % 2 )
        L++;
}
else { /* Invalid Com */
    L = 0xFF
}

return(0)

<CODE ENDS>

5.2.2. Strictly-Routed Topological Instructions

A strictly-routed topological instruction accepts the following parameters:

- An IPv6 address that identifies an interface on the segment egress node.
- A primary interface identifier.
- Zero or more secondary interface identifiers.

A strictly-routed topological instruction behaves as follows:

- If none of the interfaces identified by the above-mentioned parameters are operational, discard the packet and send an ICMPv6 Destination Unreachable message (Code: 5, Source Route Failed) to the packet’s source node.
- Overwrite the packet’s Destination Address with the IPv6 address that was received as a parameter.
- If the primary interface is active, forward the packet through the primary interface.
o If the primary interface is not active and any of the secondary interfaces are active, forward the packet through one of the secondary interfaces. Execute procedures so that all packets belonging to a flow are forwarded through the same secondary interface.

5.2.3. Loosely-Routed Topological Instructions

A loosely-routed topological instruction accepts a single parameter. This parameter is an IPv6 address that identifies an interface on the segment egress node.

A loosely-routed topological instruction behaves as follows:

o If the segment ingress node does not have a viable route to the IPv6 address included as a parameter, discard the packet and send an ICMPv6 Destination Unreachable message (Code:1 Net Unreachable) to the packet’s source node.

o Overwrite the packet’s Destination Address with the destination address that was included as a parameter.

o Forward the packet to the next hop along the least cost path to the segment egress node. If there are multiple least cost paths to the segment egress node (i.e., Equal Cost Multipath), execute procedures so that all packets belonging to a flow are forwarded through the same next hop.

6. Mutability

In the CRH, the Segments Left field is mutable. All remaining fields are immutable.

7. Compliance

In order to be compliant with this specification, an implementation MUST support 32-bit SID encoding. It MAY also support 16-bit SID encoding.

8. Management Considerations

PING and TRACEROUTE [RFC2151] both operate correctly in the presence of the CRH.
9. Security Considerations

The CRH can be used within trusted domains only. In order to enforce this requirement, domain edge routers MUST do one of the following:

- Discard all inbound packets that are destined for infrastructure interfaces and contain a CRH
- Authenticate [RFC4302] [RFC4303] all inbound packets that are destined for infrastructure interfaces and contain a CRH

10. IANA Considerations

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

<table>
<thead>
<tr>
<th>Suggested Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Compressed Routing Header (CRH)</td>
<td>This document</td>
</tr>
</tbody>
</table>

11. Acknowledgements

Thanks to Joel Halpern, Tony Li, Gerald Schmidt, Nancy Shaw and Chandra Venkatraman for their comments.

12. References

12.1. Normative References

[I-D.bonica-spring-srv6-plus]


12.2. Informative References


Appendix A. CRH Processing Examples

This appendix provides examples of CRH processing in the following applications:

- Loose source routing (Appendix A.1)
- Loose source routing preserving the first SID (Appendix A.2)
- Strict source routing (Appendix A.3)
Figure 4 provides a reference topology that is used in all examples.

<table>
<thead>
<tr>
<th>Instantiating Node</th>
<th>SID</th>
<th>Segment Type</th>
<th>IPv6 Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>1</td>
<td>Loosely-routed</td>
<td>2001:db8::1</td>
</tr>
<tr>
<td>All</td>
<td>2</td>
<td>Loosely-routed</td>
<td>2001:db8::2</td>
</tr>
<tr>
<td>All</td>
<td>3</td>
<td>Loosely-routed</td>
<td>2001:db8::3</td>
</tr>
<tr>
<td>All</td>
<td>10</td>
<td>Loosely-routed</td>
<td>2001:db8::a</td>
</tr>
<tr>
<td>All</td>
<td>11</td>
<td>Loosely-routed</td>
<td>2001:db8::b</td>
</tr>
</tbody>
</table>

Table 1: Loosely Routed SIDs

Table 1 describes SFIB entries that are instantiated on all nodes. All of these SFIB entries represent loosely-routed segments.
Table 2: Strictly Routed SIDs

Table 2 describes SFIB entries that are instantiated on specific nodes. All of these SFIB entries represent strictly-routed segments.

A.1. Loose Source Routing

In this example, Node S sends a packet to Node D, specifying loose source route through Node I3. In this example, the first node in the path, I3, does not appear in the CRH segment list. Therefore, the destination node may not be able to send return traffic through the same path.

As the packet travels from S to I3:

| Source Address = 2001:db8::a | Last Entry = 0 |
| Destination Address = 2001:db8::3 | Segments Left = 1 |
| SID[0] = 11 |

As the packet travels from I3 to D:

| Source Address = 2001:db8::a | Last Entry = 0 |
| Destination Address = 2001:db8::b | Segments Left = 0 |
| SID[0] = 11 |

A.2. Loose Source Routing Preserving The First SID

In this example, Node S sends a packet to Node D, specifying loose source route through Node I3. In this example, the first node in the path, I3, appears in the CRH segment list. Therefore, the destination node can send return traffic through the same path.
A.3. Strict Source Routing

In this example, Node S sends a packet to Node D, specifying the strict source route through I1 and I3.
As the packet travels from I3 to D:

- Source Address = 2001:db8::a
- Destination Address = 2001:db8:0:b::2
- Last Entry = 1
- Segments Left = 0
- SID[0] = 129
- SID[1] = 129

Authors’ Addresses

Ron Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, Virginia 20171
USA
Email: rbonica@juniper.net

Yuji Kamite
NTT Communications Corporation
3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan
Email: : y.kamite@ntt.com

Tomonobu Niwa
KDDI
3-22-7, Yoyogi, Shibuya-ku
Tokyo 151-0053
Japan
Email: to-niwa@kddi.com

Andrew Alston
Liquid Telecom
Nairobi
Kenya
Email: Andrew.Alston@liquidtelecom.com
Guangming Yang  
China Telecom  
109 West Zhongshan Ave, Tianhe District  
Guangzhou  
P.R. China  
Email: yanggm.gd@chinatelecom.cn

Yifeng Zhou  
ByteDance  
Building 1, AVIC Plaza, 43 N 3rd Ring W Rd Haidian District  
Beijing  
100000  
P.R. China  
Email: yifeng.zhou@bytedance.com