SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2020

R. Bonica
S. Hegde
Juniper Networks
Y. Kamite
NTT Communications Corporation
A. Alston
D. Henriques
Liquid Telecom
J. Halpern
Ericsson
J. Linkova
Google

July 3, 2019

IPv6 Support for Segment Routing: SRv6+ draft-bonica-spring-srv6-plus-01

Abstract

This document describes SRv6+. SRv6+ is a Segment Routing (SR) solution that leverages IPv6. It supports a wide variety of use-cases while remaining in strict compliance with IPv6 specifications. SRv6+ is optimized for for ASIC-based forwarding devices that operate at high data rates.

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 4, 2020.

Table of Contents

1. Overview

Network operators deploy Segment Routing (SR) [RFC8402] so that they can forward packets through SR paths. An SR path provides unidirectional connectivity from its ingress node to its egress node. While an SR path can follow the least cost path from ingress to egress, it can also follow any other path.

An SR path contains one or more segments. A segment provides unidirectional connectivity from its ingress node to its egress node. It includes a topological instruction that controls its behavior.

The topological instruction is executed on the segment ingress node. It determines the segment egress node and the method by which the segment ingress node forwards packets to the segment egress node.

Per-segment service instructions can augment a segment. Per-segment service instructions, if present, are executed on the segment egress node.

Likewise, a per-path service instruction can augment a path. The per-path service instruction, if present, is executed on the path egress node. Section 3 of this document illustrates the relationship between SR paths, segments and instructions.

A Segment Identifier (SID) identifies each segment. Because there is a one-to-one mapping between segments and the topological instructions that control them, the SID that identifies a segment also identifies the topological instruction that controls it.

A SID is different from the topological instruction that it identifies. While a SID identifies a topological instruction, it does not contain the topological instruction that it identifies. Therefore, a SID can be encoded in relatively few bits, while the topological instruction that it identifies may require many more bits for encoding.

An SR path can be represented by its ingress node as an ordered sequence of SIDs. In order to forward a packet through an SR path, the SR ingress node encodes the SR path into the packet as an ordered sequence of SIDs. It can also augment the packet with service instructions.

Because the SR ingress node is also the first segment ingress node, it executes the topological instruction associated with the first segment. This causes the packet to be forwarded to the first segment egress node. When the first segment egress node receives the packet,

it executes any per-segment service instructions that augment the first segment.

If the SR path contains exactly one segment, the first segment egress node is also the path egress node. In this case, that node executes any per-path service instruction that augments the path, and SR forwarding is complete.

If the SR path contains multiple segments, the first segment egress node is also the second segment ingress node. In this case, that node executes the topological instruction associated with the second segment. The above-described procedure continues until the packet arrives at the SR egress node.

In the above-described procedure, only the SR ingress node maintains path information. Segment ingress and egress nodes maintain information regarding the segments in which they participate, but they do not maintain path information.

The SR architecture, described above, can leverage either an MPLS [RFC3031] data plane or an IPv6 [RFC8200] data plane. SR-MPLS [I-D.ietf-spring-segment-routing-mpls] leverages MPLS. SRv6 [I-D.ietf-spring-srv6-network-programming] [I-D.ietf-6man-segment-routing-header] leverages IPv6.

This document describes SRv6+. SRv6+ is another SR variant that leverages IPv6. It supports a wide variety of use-cases while remaining in strict compliance with IPv6 specifications. SRv6+ is optimized for ASIC-based forwarding devices that operate at high data rates. Section 9 of this document highlights differences between SRv6 and SRv6+.

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. Paths, Segments And Instructions

An SRv6+ path is determined by the segments that it contains. It can be represented by its ingress node as an ordered sequence of SIDs.

A segment is determined by its ingress node and by the topological instruction that controls its behavior. The topological instruction

determines the segment egress node and the method by which the segment ingress node forwards packets to the segment egress node.

Per-segment service instructions augment, but do not determine, segments. A segment ingress node can:

- Send one packet through a segment with one per-segment service instruction.
- Send another packet through the same segment with a different per-segment service instruction.
- Send another packet through the same segment without any per-segment service instructions.

Likewise, per-path service instructions augment, but do not determine, paths.

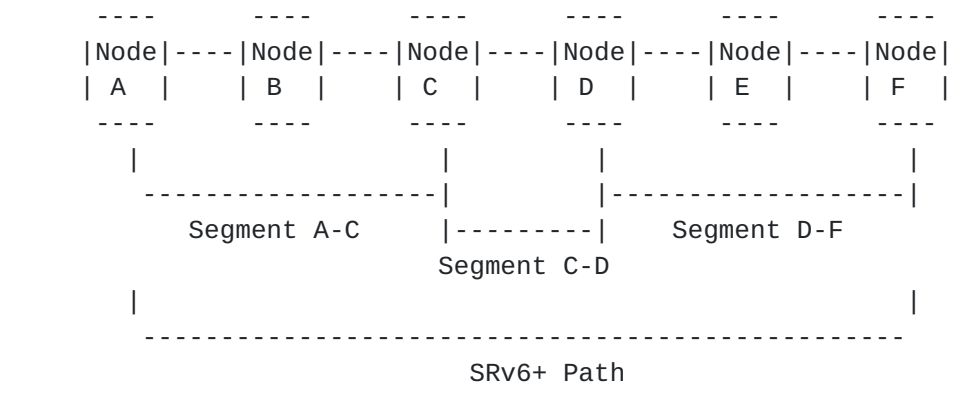

Figure 1: Paths, Segments And Instructions

Figure 1 depicts an SRv6+ path. The path provides unidirectional connectivity from its ingress node (i.e., Node A) to its egress node (i.e., Node F). It contains Segment A-C, Segment C-D and Segment D-F.

In Segment A-C, Node A is the ingress node, Node B is a transit node, and Node C is the egress node. Therefore, the topological instruction that controls the segment is executed on Node A, while per-segment service instructions that augment the segment (if any exist) are executed on Node C.

In Segment C-D, Node C is the ingress node and Node D is the egress node. Therefore, the topological instruction that controls the segment is executed on Node C, while per-segment service instructions that augment the segment (if any exist) are executed on Node D.

In Segment D-F, Node D is the ingress node, Node E is a transit node, and Node F is the egress node. Therefore, the topological instruction that controls the segment is executed on Node D, while per-segment service instructions that augment the segment (if any exist) are executed on Node F.

Node F is also the path egress node. Therefore, if a per-path service instruction augments the path, it is executed on Node F.

Segments A-C, C-D and D-F are also contained by other paths that are not included in the figure.

4. Segment Types

SRv6+ supports the following segment types:

- strictly routed
- loosely routed

Strictly routed segments forward packets through a specified link that connects the segment ingress node to the segment egress node. Loosely routed segments forward packets through the least cost path from the segment ingress node to the segment egress node.

Each segment type is described below.

4.1. Strictly Routed

When a packet is submitted to a strictly routed segment, the topological instruction associated with that segment operates upon the packet. The topological instruction executes on the segment ingress node and 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.

The 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 [RFC4443] Destination Unreachable message (Code: 5, Source Route Failed) to the packet's source node.

- Decrement the packet's Hop Count.
- If the Hop Count has expired, discard the packet and send an ICMPv6 Time Expired message 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.
- 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.

4.2. Loosely Routed

When a packet is submitted to a loosely routed segment, the topological instruction associated with that segment operates upon the packet. The topological instruction executes on the segment ingress node and accepts an IPv6 address as a parameter. The IPv6 address identifies an interface on the segment egress node.

The topological instruction behaves as follows:

- 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.
- Decrement the packet's Hop Count.
- If the Hop Count has expired, discard the packet and send an ICMPv6 Time Expired message to the packet's source node.
- Overwrite the packet's Destination Address with the destination address that was included as a parameter.
- Forward the packet to the next hop along the least cost path 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.

5. Segment Identifiers (SID)

A Segment Identifier (SID) is an unsigned integer that identifies a segment. Because there is a one-to-one mapping between segments and the topological instructions that control them, the SID that identifies a segment also identifies the topological instruction that controls it.

A SID is different from the topological instruction that it identifies. While a SID identifies a topological instruction, it does not contain the topological instruction that it identifies. Therefore, a SID can be encoded in relatively few bits, while the topological instruction that it identifies may require many more bits for encoding.

SIDs have node-local significance. This means that a segment ingress node MUST identify each segment that it originates with a unique SID. However, a SID that is used by one segment ingress node to identify a segment that it originates can be used by another segment ingress node to identify another segment.

Although SIDs have node-local significance, an SRv6+ path can be uniquely identified by its ingress node and an ordered sequence of SIDs. This is because the topological instruction associated with each segment determines the ingress node of the next segment (i.e., the node upon which the next SID has significance.)

Although SIDs have node-local significance, they can be assigned in a manner that facilitates debugging. See Section 5.2 and Section 5.3 for details.

5.1. Range

SID values range from 0 to a configurable Maximum SID Value (MSV). The values 0 through 15 are reserved for future use. The following are valid MSVs:

- 65,535 (i.e., 2**16 minus 1)

- 4,294,967,295 (i.e., 2**32 minus 1)

In order to optimize packet encoding (Section 7.1), network operators can configure all nodes within an SRv6+ domain to have the smallest feasible MSV. The following paragraphs explain how an operator determines the smallest feasible MSV.

Consider an SRv6+ domain that contains 5,000 nodes connected to one another by point-to-point infrastructure links. The network topology

is not a full-mesh. In fact, each node supports 200 point-to-point infrastructure links or fewer. Given this SRv6+ domain, we will determine the smallest feasible MSV under the following conditions:

- The SRv6+ domain contains strictly routed segments only.
- The SRv6+ domain contains loosely routed segments only.
- The SRv6+ domain contains both strictly and loosely routed segments.

If an SRv6+ domain contains strictly routed segments only, and each node creates a strictly routed segment to each of its neighbors, each node will create 200 segments or fewer and consume 200 SIDs or fewer. This is because each node has 200 neighbors or fewer. Because SIDs have node-local significance (i.e., they can be reused across nodes), the smallest feasible MSV is 65,535.

Adding nodes to this SRv6+ domain will not increase the smallest feasible MSV, so long as each node continues to support 65,519 point-to-point infrastructure links or fewer. If a single node is added to the domain and that node supports 240 infrastructure links, the smallest feasible MSV will increase to 65,535.

If an SRv6+ domain contains loosely routed segments only, and every node creates a loosely routed segment to every other node, every node will create 4,999 segments and consume 4,999 SIDs. This is because the domain contains 5,000 nodes. Because SIDs have node-local significance (i.e., they can be reused across nodes), the smallest feasible MSV is 65,535.

Adding nodes to this SRv6+ domain will not increase the smallest feasible MSV until the number of nodes exceeds 65,519. When the smallest feasible MSV increases, it becomes 4,294,967,295.

If an SRv6+ domain contains both strictly and loosely routed segments, each node will create 5,199 segments or fewer and consume 5,199 SIDs or fewer. This value is the sum of the following:

- The number of loosely routed segments that each node will create, given that every node creates a loosely routed segment to every other node (i.e., 4,999).
- The number of strictly routed segments that each node will create, given that each node creates a strictly routed segment to each of its neighbors (i.e., 200 or fewer).

Because SIDs have node-local significance (i.e., they can be reused across nodes), the smallest feasible MSV is 65,535.

Adding nodes to this SRv6+ domain will not increase the smallest feasible MSV until the number of nodes plus the maximum number of infrastructure links per node exceeds 65,519. When the smallest feasible MSV increases, it becomes 4,294,967,295.

5.2. Assigning SIDs to Strictly Routed Segments

Network operators can establish conventions by which they assign SIDs to strictly routed segments. These conventions can facilitate debugging.

For example, a network operator can reserved a range of SIDs for strictly routed segments. It can further divide that range into subranges, so that all segments sharing a common egress node are identified by SIDs from the same subrange.

5.3. Assigning SIDs to Loosely Routed Segments

In order to facilitate debugging, all loosely routed segments that share a common egress node are identified by the same SID. In order to maintain this discipline, network wide co-ordination is required.

For example, assume that an SRv6+ domain contains N nodes. Network administrators reserve a block of N SIDs and configure one of those SIDs on each node. Each node advertises its SID into the control plane. When another node receives that advertisement, it creates a loosely routed segment between itself and the advertising node. It also associates the SID that it received in the advertisement with the newly created segment. See [I-D.bonica-lsr-crh-isis-extensions] for details.

6. Service Instructions

SRv6+ supports the following service instruction types:

- Per-segment
- Per-path

Each is described below.

6.1. Per-Segment

Per-segment service instructions can augment a segment. Per-segment service instructions, if present, are executed on the segment egress node. Because the path egress node is also a segment egress node, it can execute per-segment service instructions.

The following are examples of per-segment service instructions:

- Expose a packet to a firewall policy.

- Expose a packet to a sampling policy.

Per-segment Service Instruction Identifiers identify a set of service instructions. Per-segment Service Instruction Identifiers are allocated and distributed by a controller. They have domain-wide significance.

6.2. Per-Path

A per-path service instruction can augment a path. The per-path service instruction, if present, is executed on the path egress node.

The following are examples of per-path service instructions:

- De-encapsulate a packet and forward its newly exposed payload through a specified interface.

- De-encapsulate a packet and forward its newly exposed payload using a specified routing table.

Per-path Service Instruction Identifiers identify per-path service instructions. Per-path Service Instruction Identifiers are allocated and distributed by the processing node (i.e., the path egress node). They have node-local significance. This means that the path egress node MUST allocate a unique Per-path Service Instruction Identifier for each per-path service instruction that it instantiates.

7. The IPv6 Data Plane

SRv6+ ingress nodes generate IPv6 header chains that represent SRv6+ paths. An IPv6 header chain contains an IPv6 header. It can also contain one or more extension headers.

An extension header chain that represents an SRv6+ path can contain any valid combination of IPv6 extension headers. The following bullet points describe how SRv6+ leverages IPv6 extension headers:

- If an SRv6+ path contains multiple segments, the IPv6 header chain that represents it MUST contain a Routing header. The SRv6+ path MUST be encoded in the Routing header as an ordered sequence of SIDs.

- If an SRv6+ path is augmented by a per-path service instruction, the IPv6 header chain that represents it MUST contain a Destination Options header. The Destination Options header MUST immediately precede an upper-layer header and it MUST include a Per-Path Service Instruction Identifier.

- If an SRv6+ path contains a segment that is augmented by a per-segment service instruction, the IPv6 chain that represents it MUST contain a Routing header and a Destination Options header. The Destination Options header MUST immediately precede a Routing header and it MUST include the Per-Segment Service Instruction Identifier.

The following subsections describe how SRv6+ uses the Routing header and the Destination Options header.

7.1. The Routing Header

SRv6+ defines a new Routing header type, called the Compressed Routing Header (CRH) [I-D.bonica-6man-comp-rtg-hdr]. The CRH contains the following fields:

- Next Header - Identifies the header immediately following the CRH.

- Hdr Ext Len - Length of the CRH.

- Routing Type - Identifies the Routing header variant (i.e., CRH)

- Segments Left - The number of segments still to be traversed before reaching the path egress node.

- Last Entry - Represents the index of the last element of the Segment List.

- Com (Compression) - 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 MSV (Section 5.1).

- 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.

As per [RFC8200], when an IPv6 node receives a packet, it examines the packet's destination address. If the destination address represents an interface belonging to the node, the node processes the next header. If the node encounters and recognizes the CRH, it processes the CRH as follows:

- If Segments Left equal 0, skip over the CRH and process the next header in the packet.

- Decrement Segments Left.

- Search for the current SID in a local table that maps SID's to topological instructions. If the current SID cannot be found in that table, send an ICMPv6 Parameter Problem message to the packet's Source Address and discard the packet.

- Execute the topological instruction found in the table as described in Section 4. This causes the packet to be forwarded to the segment egress node.

When the packet arrives at the segment egress node, the above-described procedure is repeated.

7.2. The Destination Options Header

According to [RFC8200], the Destination Options header contains one or more IPv6 options. It can occur twice within a packet, once before a Routing header and once before an upper-layer header. The Destination Options header that occurs before a Routing header is processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations that are listed in the Routing header. The Destination Options header that occurs before an upper-layer header is processed by the packet's final destination only.

Therefore, SRv6+ defines the following new IPv6 options:

- The SRv6+ Per-Segment Service Instruction Option [I-D.bonica-6man-seg-end-opt]

- The SRv6+ Per-Path Service Instruction Option [I-D.bonica-6man-vpn-dest-opt]

The SRv6+ Per-Segment Service Instruction Option is encoded in a Destination Options header that precedes the CRH. Therefore, it is

processed by every segment egress node. It includes a Per-Segment Service Instruction Identifier and causes segment egress nodes to execute per-segment service instructions.

The SRv6+ Per-Path Service Instruction Option is encoded in a Destination Options header that precedes the upper-layer header. Therefore, it is processed by the path egress node only. It includes a Per-Path Service Instruction Identifier and causes the path egress node to execute a per-path service instruction.

8. Control Plane

IS-IS extensions [I-D.bonica-lsr-crh-isis-extensions] have been defined for the following purposes:

- So that SRv6+ segment ingress nodes can flood information regarding strictly routed segments that they originate

- So that SRv6+ segment egress nodes can flood information regarding loosely routed segments that they terminate

BGP extensions [RFC4271] are being defined so that SRv6+ path egress nodes can associated path-terminating service instructions with Network Layer Reachability Information (NLRI).

9. Differences Between SRv6 and SRv6+

9.1. Routing Header Size

SRv6 defines a Routing header type, called the Segment Routing Header (SRH). The SRH contains a field that represents the SRv6 path as an ordered sequence of SIDs. Each SID contained by that field is 128 bits long.

Likewise, SRv6+ defines a Routing Header Type, called the Compressed Routing Header (CRH). The CRH contains a field that represents the SRv6+ path as an ordered sequence of SIDs. Within that field, SIDs can be 16 or 32 bits long.

SIDs	SRv6 SRH (128-bit SID)	SRv6+ CRH (16-bit SID)	SRv6+ CRH (32-bit SID)
1	24	16	16
2	40	16	16
3	56	16	24
4	72	16	24
5	88	24	32
6	104	24	32
7	120	24	40
8	136	24	40
9	152	32	48
10	168	32	48
11	184	32	56
12	200	32	56
13	216	40	64
14	232	40	64
15	248	40	72
16	264	40	72

Table 1: Routing Header Size (in Bytes) As A Function Of Routing Header Type and Number Of SIDs

Table 1 reflects Routing header size as a function of Routing header type and Number of SIDs contained by the Routing header.

Large Routing headers are undesirable for the following reasons:

- Many ASIC-based forwarders copy the entire IPv6 extension header chain from buffer memory to on-chip memory. As the size of the IPv6 extension header chain increases, so does the cost of this copy.

- Because Path MTU Discovery (PMTUD) [RFC8201] is not entirely reliable, many IPv6 hosts refrain from sending packets larger than the IPv6 minimum link MTU (i.e., 1280 bytes). When packets are small, the overhead imposed by large Routing headers becomes pronounced.

9.2. Decoupling of Topological and Service Instructions

SRv6+ decouples topological instructions from service instructions. Topological instructions are invoked at the segment ingress node, as a result of CRH processing, while service instructions are invoked at the segment egress node, as a result of Destination Option

processing. Therefore, network operators can use SRv6+ mechanisms to support topological instructions, service instructions, or both.

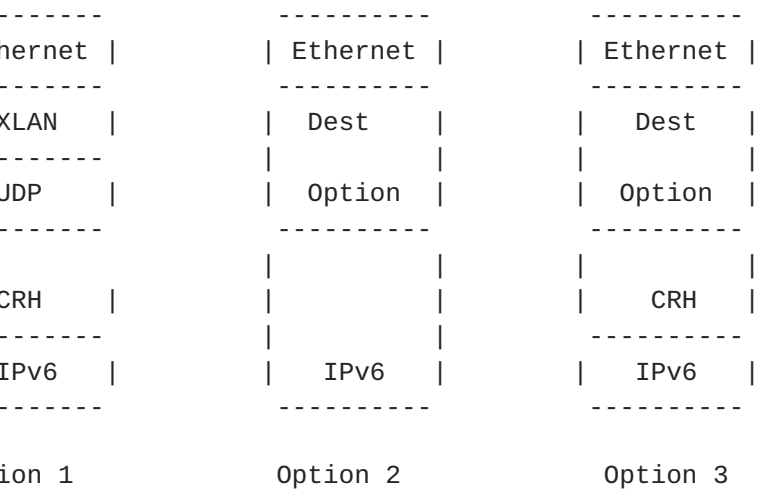

Figure 2: EVPN Design Alternatives

Figure 2 illustrates this point by depicting design options available to network operators offering Ethernet Virtual Private Network [RFC7432] services over Virtual eXtensible Local Area Network (VXLAN) [RFC7348]. In Option 1, the network operator encodes topological instructions in the CRH, while encoding service instructions in a VXLAN header. In Option 2, the network operator encodes service instructions in a Destination Options header, while allowing traffic to traverse the least cost path between the ingress and egress Provider Edge (PE) routers. In Option 3, the network operator encodes topological instructions in the CRH, and encodes service instructions in a Destination Options header.

9.3. Authentication

The IPv6 Authentication Header (AH) [RFC4302] can be used to authenticate SRv6+ packets. However, AH processing is not defined in SRv6.

9.4. Traffic Engineering Capability

SRv6+ supports traffic engineering solutions that rely exclusively upon strictly routed segments. For example, consider an SRv6+ network whose diameter is 12 hops and whose minimum feasible MSV is 65,525. In that network, in the worst case, SRv6+ overhead is 72 bytes (i.e., a 40-byte IPv6 header and a 32-byte CRH).

SRv6 also supports traffic engineering solutions that rely exclusively upon strictly routed segments (i.e., END.X SIDs).

However, SRv6 overhead may be prohibitive. For example, consider an SRv6 network whose diameter is 12 hops. In the worst case, SRv6 overhead is 240 bytes (i.e., a 40 byte IPv6 header and a 200-byte SRH).

9.5. IP Addressing Architecture

In SRv6, an IPv6 address can represent either of the following:

- A network interface
- An instruction instantiated on a node (i.e., an SRv6 SID)

In SRv6+ an IPv6 address always represents a network interface, as per [RFC4291].

10. Compliance

In order to be compliant with this specification, an SRv6+ implementation MUST:

- Be able to process IPv6 options as described in Section 4.2 of [RFC8200].
- Be able to process the Routing header as described in Section 4.4 of [RFC8200].
- Be able to process the Destination Options header as described in Section 4.6 of [RFC8200].
- Recognize the CRH.
- Be able to encode an SRv6+ path in the CRH as an ordered sequence of 32-bit SIDs.
- Be able to process a CRH that includes 32-bit SIDs.

Additionally, an SRv6+ implementation MAY:

- Be able to encode an SRv6+ path in the CRH as an ordered sequence of 16-bit SIDs.
- Be able to process a CRH that includes 16-bit SIDs.
- Recognize the Per-Segment Service Instruction Option.
- Recognize the Per-Path Service Instruction Option.

11. Operational Considerations

11.1. Ping and Traceroute

Ping and Traceroute [RFC2151] both operate correctly in SRv6+ (i.e., in the presence of the CRH).

11.2. ICMPv6 Rate Limitting

As per [RFC4443], SRv6+ nodes rate limit the ICMPv6 messages that they emit.

11.3. SID Lengths And SID Length Transitions

An SRv6+ implementation MAY include a configuration option that determines how it encodes SIDs (i.e., in 16 or 32 bits). In order to reduce operational complexity, network operators typically configure their networks so that every node encodes SIDs identically.

As a network grows, its minimum feasible MSV may increase. In this case, the network may need to migrate from one SID encoding to another. The following bullet points describe a migration strategy for an SRv6+ network that is migrating from 16-bit SIDs to 32-bit SIDs:.

- Ensure that all nodes can process a CRH that includes 32-bit SIDs.
- Configure each nodes so that encodes SIDs in 32-bits.
- Configure SIDs whose value exceeds 65,535.

12. IANA Considerations

SID values 0-15 are reserved for future use. They may be assigned by IANA, based on IETF Consensus.

IANA is requested to establish a "Registry of SRv6+ Reserved SIDs". Values 0-15 are reserved for future use.

13. Security Considerations

SRv6+ domains MUST NOT span security domains. In order to enforce this requirement, security domain edge routers MUST do one of the following:

- Discard all inbound SRv6+ packets
- Authenticate [RFC4302] [RFC4303] all inbound SRv6+ packets

14. Acknowledgements

The authors wish to acknowledge Dr. Vanessa Ameen and John Scudder.

15. References

15.1. Normative References

[I-D.bonica-6man-comp-rtg-hdr]
Bonica, R., Kamite, Y., Niwa, T., Alston, A., Henriques, D., So, N., Xu, F., Chen, G., Zhu, Y., Yang, G., and Y. Zhou, "The IPv6 Compressed Routing Header (CRH)", draft-bonica-6man-comp-rtg-hdr-04 (work in progress), May 2019.

[I-D.bonica-6man-seg-end-opt]
Bonica, R., Halpern, J., So, N., Xu, F., Chen, G., Zhu, Y., Yang, G., and Y. Zhou, "The IPv6 Segment Endpoint Option", draft-bonica-6man-seg-end-opt-03 (work in progress), March 2019.

[I-D.bonica-6man-vpn-dest-opt]
Bonica, R., Lenart, C., So, N., Xu, F., Presbury, G., Chen, G., Zhu, Y., Yang, G., and Y. Zhou, "The IPv6 Virtual Private Network (VPN) Context Information Option", draft-bonica-6man-vpn-dest-opt-05 (work in progress), March 2019.

[I-D.bonica-lsr-crh-isis-extensions]
Kaneriya, P., Shetty, R., Hegde, S., and R. Bonica, "IS-IS Extensions To Support The IPv6 Compressed Routing Header (CRH)", draft-bonica-lsr-crh-isis-extensions-00 (work in progress), May 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

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

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

15.2. Informative References

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-21 (work in progress), June 2019.

[I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-22 (work in progress), May 2019.

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-00 (work in progress), April 2019.

[RFC2151] Kessler, G. and S. Shepard, "A Primer On Internet and TCP/IP Tools and Utilities", FYI 30, RFC 2151, DOI 10.17487/RFC2151, June 1997, <https://www.rfc-editor.org/info/rfc2151>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

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

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

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <https://www.rfc-editor.org/info/rfc4761>.

[RFC6624] Kompella, K., Kothari, B., and R. Cherukuri, "Layer 2 Virtual Private Networks Using BGP for Auto-Discovery and Signaling", RFC 6624, DOI 10.17487/RFC6624, May 2012, <https://www.rfc-editor.org/info/rfc6624>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/info/rfc7348>.

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <https://www.rfc-editor.org/info/rfc7432>.

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

Authors' Addresses

Ron Bonica
Juniper Networks
Herndon, Virginia 20171
USA

Email: rbonica@juniper.net

Shraddha Hegde
Juniper Networks
Embassy Business Park
Bangalore, KA 560093
India

Email: shraddha@juniper.net

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

Email: : y.kamite@ntt.com

Andrew Alston
Liquid Telecom
Nairobi
Kenya

Email: Andrew.Alston@liquidtelecom.com

Daniam Henriques
Liquid Telecom
Johannesburg
South Africa

Email: daniam.henriques@liquidtelecom.com

Joel Halpern
Ericsson
P. O. Box 6049
Leesburg, Virginia 20178
USA

Email: joel.halpern@ericsson.com

Jen Linkova
Google
Mountain View, California 94043
USA

Email: furry@google.com