SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 30, 2018

Z. Ali
C. Filsfils
N. Kumar
C. Pignataro
F. Iqbal
R. Gandhi
Cisco Systems, Inc.
J. Leddy
Comcast
S. Matsushima
SoftBank
R. Raszuk
Bloomberg LP
B. Peirens
Proximus
G. Naik
Drexel University
February 26, 2018

Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)
draft-ali-spring-srv6-oam-00.txt

Abstract

This document describes mechanisms for Operations, Administration, and Maintenance (OAM) in Segment Routing with IPv6 data plane (SRv6) network.

Status of This Memo

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

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

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

Copyright Notice

Table of Contents

1. Introduction

This document describes mechanisms for Operations, Administrations, and Maintenance (OAM) in Segment Routing using IPv6 data plane (SRv6) networks.

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

2. Conventions Used in This Document

2.1. Abbreviations

ECMP: Equal Cost Multi-Path.

SID: Segment ID.

SL: Segment Left.

SR: Segment Routing.

SRH: Segment Routing Header.

SRv6: Segment Routing with IPv6 Data plane.

TC: Traffic Class.

UCMP: Unequal Cost Multi-Path.

2.2. Terminology and Reference Topology

In this document, the simple topology shown in Figure 1 is used for illustration.

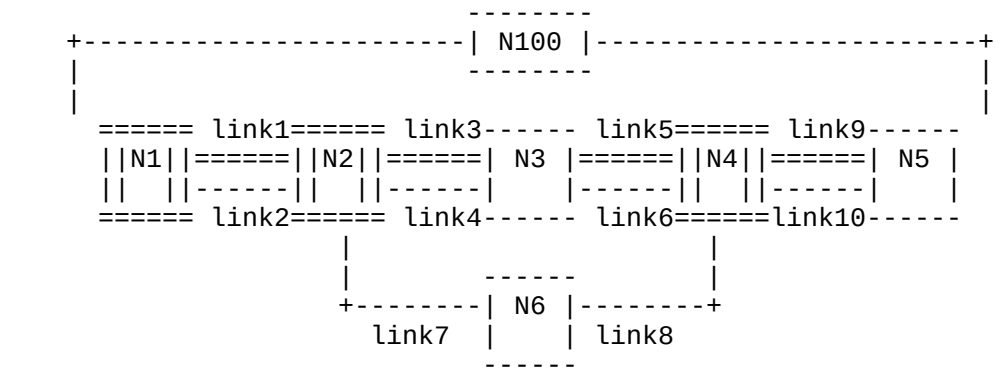


Figure 1: Reference Topology

In the reference topology:

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

Nodes N3, N5 and N6 are classic IPv6 nodes.

Node 100 is a controller.

Node Nk has a classic IPv6 loopback address Bk::/128

Node Nk has Ak::/48 for its local SID space from which Local SIDs are explicitly allocated.

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

Ak::0 is explicitly allocated as the END function at Node k.

Ak::Cij is explicitly allocated as the END.X function at node k towards neighbor node i via jth Link between node i and node j. e.g., A2::C31 represents END.X at N2 towards N3 via link3 (the 1st link between N2 and N3). Similarly, A4::C52 represents the END.X at N4 towards N5 via link10.

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

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

SR policy is defined in Section 3 of [I-D.spring-segment-routing-policy].

3. OAM Mechanisms

This section describes how ping and traceroute mechanisms can be used in an SRv6 network. Additional OAM mechanisms will be added in a future revision of the document.

3.1. Ping

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

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

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

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

3.1.1. Classic Ping

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

If an SRv6 capable ingress node wants to ping an IPv6 prefix via an arbitrary segment list <S1, S2, S3>, it needs to initiate ICMPv6 ping with an SR header containing the SID list <S1, S2, S3>. This is illustrated using the topology in Figure 1. Assume all the links have IGP metric 10 except both links between node N2 and node N3, which have IGP metric set to 100. User issues a ping from node N1 to a loopback of node N5, via via segment list <A2::C31, A4::C52>.

Figure 2 contains sample output for a ping request initiated at node N1 to the loopback address of node N5 via a segment list <A2::C31, A4::C52>.

```
> ping B5:: via segment-list A2::C31, A4::C52

Sending 5, 100-byte ICMP Echos to B5::, timeout is 2 seconds:
```

```
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 0.625
/0.749/0.931 ms
```

Figure 2: A sample ping output at an SRv6 capable node

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

- Node N1 initiates an ICMPv6 ping packet with SRH as follows (B1::,A2::C31)(B1::, A4::C52, A2::C31, SL=2, NH: ICMPv6)(ICMPv6 Echo Request).
- Node N2, which is an SRv6 capable node, performs the standard SRH processing. Specifically, it executes the END.X function (A2::C31) on the echo request packet.
- Node N3, which is a classic IPv6 node, performs the standard IPv6 processing. Specifically, it forwards the echo request based on DA A4::C52 in the IPv6 header.
- Node N4, which is an SRv6 capable node, performs the standard SRH processing. Specifically, it observes the END.X function (A4::C52) with PSP (Penultimate Segment Popping) on the echo request packet and removes the SRH and forwards the packet across link10 to N5.
- The echo request packet at N5 arrives as an IPv6 packet without a SRH. Node N5, which is a classic IPv6 node, performs the standard IPv6/ICMPv6 processing on the echo request and responds, accordingly.

3.1.2. Pinging SID Function

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

Consider the case where the user wants to ping the remote SID function A4::C52, via A2::C31, from node N1. Node N1 constructs the ping packet (B1::0, A2::C31)(A4::C52, A2::C31, SL=1;NH=ICMPv6)(ICMPv6 Echo Request). When the node N4 receives the ICMPv6 echo request with DA set to A4::C52 and next header set to ICMPv6, it silently drops it (as per [I-D.filsfils-spring-srv6-network-programming]). To solve this

problem, the initiator needs to mark the ICMPv6 echo request as an OAM packet.

The OAM packets are identified either by setting the O-bit in SRH [I-D.6man-segment-routing-header] or by inserting the SID Function END.OTP at an appropriate place in the SRH [I-D.filsfils-spring-srv6-network-programming].

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

3.1.2.1. End-to-end Ping Using END.OTP

Consider the same example where the user wants to ping a remote SID function A4::C52 , via A2::C31, from node N1. To force a punt of the ICMPv6 echo request at the node N4, node N1 inserts the SID function END.OTP just before the target SID A4::C52 in the SRH. The ICMPv6 echo request is processed at the individual nodes along the path as follows:

- Node N1 initiates an ICMPv6 ping packet with SRH as follows (B1::0, A2::C31)(A4::C52, A4::OTP, A2::C31; SL=2; NH=ICMPv6)(ICMPv6 Echo Request).

- Node N2, which is an SRv6 capable node, performs the standard SRH processing. Specifically, it executes the END.X function (A2::C31) on the echo request packet.

- Node N3 receives the packet as follows (B1::0, A4::OTP)(A4::C52, A4::OTP, A2::C31 ; SL=1; NH=ICMPv6)(ICMPv6 Echo Request). Node N3, which is a classic IPv6 node, performs the standard IPv6 processing. Specifically, it forwards the echo request based on DA A4::OTP in the IPv6 header.

- When node N4 receives the packet (B1::0, A4::OTP)(A4::C52,A4::OTP, A2::C31 ; SL=1; NH=ICMPv6)(ICMPv6 Echo Request), it processes the SID Function END.OTP, as described in the pseudocode in [I-D.filsfils-spring-srv6-network-programming]. The packet gets punted to the ICMPv6 process for processing. The ICMPv6 process checks if the next SID in SRH (the target SID A4::C52) is locally programmed.

- If the target SID is not locally programmed, N4 responses with the ICMPv6 message (Type: "SRv6 OAM (TBA1 by IANA)", Code: "SID not locally implemented (TBA2 by IANA)"); otherwise a success is returned.

3.1.2.2. Segment-by-segment Ping Using O-bit (Proof of Transit)

Consider the same example where the user wants to ping a remote SID function A4::C52 , via A2::C31, from node N1. However, in this ping, the node N1 wants to get a response from each segment node in the SRH. In other words, in the segment-by-segment ping case, the node N1 expects a response from node N2 and node N4 for their respective local SID function.

To force a punt of the ICMPv6 echo request at node N2 and node N4, node N1 sets the O-bit in SRH [I-D.6man-segment-routing-header]. The ICMPv6 echo request is processed at the individual nodes along the path as follows:

- Node N1 initiates an ICMPv6 ping packet with SRH as follows (B1::0, A2::C31)(A4::C52, A2::C31; SL=1, Flags.O=1; NH=ICMPv6)(ICMPv6 Echo Request).

- When node N2 receives the packet (B1::0, A2::C31)(A4::C52, A2::C31; SL=1, Flags.O=1; NH=ICMPv6)(ICMPv6 Echo Request) packet, it processes the O-bit in SRH, as described in the pseudo code in [I-D.filsfils-spring-srv6-network-programming]. A time-stamped copy of the packet is punted to the ICMPv6 process in control plane for processing. Node N2 continues to apply the A2::C31 SID function on the original packet and forwards it, accordingly. Due to SRH.Flags.O=1, Node N2 also disables the PSP behaviour, i.e., does not remove the SRH. The ICMPv6 process at node N2 checks if its local SID (A2::C31) is locally programmed or not and responds to the ICMPv6 Echo Request.

- If the target SID is not locally programmed, N4 responses with the ICMPv6 message (Type: "SRv6 OAM (TBA1 by IANA)", Code: "SID not locally implemented (TBA2 by IANA)"); otherwise a success is returned. Note that, as mentioned in [I-D.filsfils-spring-srv6-network-programming], if node N2 does not support the O-bit, it simply ignores it and process the local SID, A2::C31.

- Node N3, which is a classic IPv6 node, performs standard IPv6 processing. Specifically, it forwards the echo request based on DA A4::C52 in the IPv6 header.

- When node N4 receives the packet (B1::0, A4::C52)(A4::C52, A2::C31; SL=0, Flags.O=1; NH=ICMPv6)(ICMPv6 Echo Request), it processes the O-bit in SRH, as described in the pseudo code in [I-D.filsfils-spring-srv6-network-programming]. A time-stamped copy of the packet is punted to the ICMPv6 process in control plane for processing. The ICMPv6 process at node N4 checks if its

local SID (A2::C31) is locally programmed or not and responds to the ICMPv6 Echo Request.

If the target SID is not locally programmed, N4 responses with the ICMPv6 message (Type: "SRv6 OAM (TBA1 by IANA)", Code: "SID not locally implemented (TBA2 by IANA)"); otherwise a success is returned.

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

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

3.2. Error Reporting

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

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

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

3.3. Traceroute

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

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

3.3.1. Classic Traceroute

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

If an SRv6 capable ingress node wants to traceroute to IPv6 prefix via an arbitrary segment list <S1, S2, S3>, it needs to initiate traceroute probe with an SR header containing the SID list <S1, S2, S3>. This is illustrated using the topology in Figure 1. Assume all the links have IGP metric 10 except both links between node N2 and node N3, which have IGP metric set to 100. User issues a traceroute from node N1 to a loopback of node N5, via segment list <A2::C31, A4::C52>. Figure 3 contains sample output for the traceroute request.

```
> traceroute B5:: via segment-list A2::C31, A4::C52

Tracing the route to B5::

 1  2001:DB8:1:2:21:: 0.512 msec 0.425 msec 0.374 msec
    SRH: (B5::, A4::C52, A2::C31, SL=2)

 2  2001:DB8:2:3:31:: 0.721 msec 0.810 msec 0.795 msec
    SRH: (B5::, A4::C52, A2::C31, SL=1)

 3  2001:DB8:3:4:41:: 0.921 msec 0.816 msec 0.759 msec
    SRH: (B5::, A4::C52, A2::C31, SL=1)

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

Figure 3: A sample traceroute output at an SRv6 capable node

Please note that information for hop2 is returned by N3, which is a classic IPv6 node. Nonetheless, the ingress node is able to display

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

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

There is no difference in processing of the traceroute probe at an IPv6 classic node and an SRv6 capable node. Similarly, both IPv6 classic and SRv6 capable nodes use the address of the interface on which probe was received as the source address in the ICMPv6 response. ICMP extensions defined in [RFC5837] can be used to also display information about the IP interface through which the datagram would have been forwarded had it been forwardable, and the IP next hop to which the datagram would have been forwarded, the IP interface upon which a datagram arrived, the sub-IP component of an IP interface upon which a datagram arrived.

The information about the IP address of the incoming interface on which the traceroute probe was received by the reporting node is very useful. This information can also be used to verify if SID functions A2::C31 and A4::C52 are executed correctly by N2 and N4, respectively. Specifically, the information displayed for hop2 contains the incoming interface address 2001:DB8:2:3::31 at N3. This matches with the expected interface bound to END.X function A2::C31 (link3). Similarly, the information displayed for hop5 contains the incoming interface address 2001:DB8:4:5::52 at N5. This matches with the expected interface bound to the END.X function A4::C52 (link10).

3.3.2. Traceroute to a SID Function

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

Consider the case where the user wants to traceroute the remote SID function A4::C52, via A2::C31, from node N1. Node N1 constructs the traceroute packet (B1::0, A2::C31, HC=1) (A4::C52, A2::C31, SL=1; NH=UDP) (traceroute probe). Even though Hop Count of the packet is set to 1, when the node N4 receives the traceroute probe with DA set to A4::C52 and next header set to UDP, it silently drops it (as per

[I-D.filsfils-spring-srv6-network-programming]). To solve this problem, the initiator node needs to mark the traceroute probe as an OAM packet.

The OAM packets are identified either by setting the O-bit in SRH [I-D.6man-segment-routing-header] or by inserting the SID Function END.OTP at an appropriate place in the SRH [I-D.filsfils-spring-srv6-network-programming].

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

- o In hop-by-hop traceroute, user gets responses from all nodes including classic IPv6 transit nodes, SRv6 capable transit nodes as well as SRv6 capable segment endpoints. E.g., consider the example where the user wants to traceroute to a remote SID function A4::C52, via A2::C31, from node N1. The traceroute output will also display information about node N3, which is a transit (underlay) node.

- o The overlay traceroute, on the other hand, does not trace the underlay nodes. In other words, the overlay traceroute only displays the nodes that acts as SRv6 segments along the route. I.e., in the example where the user wants to traceroute to a remote SID function A4::C52, via A2::C31, from node N1, the overlay traceroute would only display the traceroute information from node N2 and node N4 and will not display information from node N3.

3.3.2.1. Hop-by-hop Traceroute Using END.OTP

In this Section, hop-by-hop traceroute to a SID function is exemplified using UDP probes. However, the procedure is equally applicable to other implementation of traceroute mechanism.

Consider the same example where the user wants to traceroute to a remote SID function A4::C52 , via A2::C31, from node N1. To force a punt of the traceroute probe only at the node N4, node N1 inserts the SID Function END.OTP just before the target SID A4::C52 in the SRH. The traceroute probe is processed at the individual nodes along the path as follows:

- o Node N1 initiates a traceroute probe packet with a monotonically increasing value of hop count and SRH as follows (B1::0,A2::C31)(A4::C52, A4::OTP, A2::C31; SL=2; NH=UDP)(Traceroute probe).

- o When node N2 receives the packet with hop-count = 1, it processes

the hop count expiry. Specifically, the node N2 responses with the ICMPv6 message (Type: "Time Exceeded", Code: "Time to Live exceeded in Transit").

- When Node N2 receives the packet with hop-count > 1, it performs the standard SRH processing. Specifically, it executes the END.X function (A2::C31) on the traceroute probe.

- When node N3, which is a classic IPv6 node, receives the packet (B1::0, A4::OTP)(A4::C52, A4::OTP, A2::C31 ; HC=1, SL=1; NH=UDP)(Traceroute probe) with hop-count = 1, it processes the hop count expiry. Specifically, the node N3 responses with the ICMPv6 message (Type: "Time Exceeded", Code: "Time to Live exceeded in Transit").

- When node N3, which is a classic IPv6 node, receives the packet with hop-count > 1, it performs the standard IPv6 processing. Specifically, it forwards the traceroute probe based on DA A4::OTP in the IPv6 header.

- When node N4 receives the packet (B1::0, A4::OTP)(A4::C52, A4::OTP, A2::C31 ; SL=1; HC=1, NH=UDP)(Traceroute probe), it processes the SID Function END.OTP, as described in the pseudocode in [I-D.filsfils-spring-srv6-network-programming]. The packet gets punted to the traceroute process for processing. The traceroute process checks if the next SID in SRH (the target SID A4::C52) is locally programmed. If the target SID A4::C52 is locally programmed, node N4 responses with the ICMPv6 message (Type: Destination unreachable, Code: Port Unreachable). If the target SID A4::C52 is not a local SID, node N4 silently drops the traceroute probe.

Figure 4 displays a sample traceroute output for this example.

```
> traceroute srv6 A4::C52 via segment-list A2::C31

Tracing the route to SID function A4::C52

 1  2001:DB8:1:2::21 0.512 msec 0.425 msec 0.374 msec   SRH:
 (A4::C52, A4::OTP, A2::C31; SL=2)

 2  2001:DB8:2:3::31 0.721 msec 0.810 msec 0.795 msec   SRH:
 (A4::C52, A4::OTP, A2::C31; SL=1)

 3  2001:DB8:3:4::41 0.921 msec 0.816 msec 0.759 msec   SRH:
 (A4::C52, A4::OTP, A2::C31; SL=1)
```

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

3.3.2.2. Tracing SRv6 Overlay

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

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

Consider the same example where the user wants to traceroute to a remote SID function A4::C52 , via A2::C31, from node N1.

- Node N1 initiates a traceroute probe with SRH as follows (B1::0,A2::C31)(A4::C52, A2::C31; HC=64, SL=1, Flags.O=1; NH=UDP)(Traceroute Probe). Please note that the hop-count is set to 64 to skip the underlay nodes from tracing. The O-bit in SRH is set to make the overlay nodes (nodes processing the SRH) respond.

- When node N2 receives the packet (B1::0, A2::C31)(A4::C52,A2::C31; SL=1, HC=64, Flags.O=1; NH=UDP)(Traceroute Probe), it processes the O-bit in SRH, as described in the pseudocode in [I-D.filsfils-spring-srv6-network-programming]. A time-stamped copy of the packet gets punted to the traceroute process for processing. Node N2 continues to apply the A2::C31 SID function on the original packet and forwards it, accordingly. As SRH.Flags.O=1, Node N2 also disables the PSP flavor, i.e., does not remove the SRH. The traceroute process at node N2 checks if its local SID (A2::C31) is locally programmed. If the SID is not locally programmed, it silently drops the packet. Otherwise, it performs the egress check by looking at the SL value in SRH.

- As SL is not equal to zero (i.e., it’s not egress node), node N2 responses with the ICMPv6 message (Type: "SRv6 OAM (TBA1 by IANA)", Code: "O-bit punt at Transit (TBA3 by IANA)"). Note that, as mentioned in [I-D.filsfils-spring-srv6-network-programming], if node N2 does not support the O-bit, it simply ignores it and processes the local SID, A2::C31.

- When node N3 receives the packet (B1::0, A4::C52)(A4::C52, A2::C31; SL=0, HC=63, Flags.O=1; NH=UDP)(Traceroute Probe), performs the standard IPv6 processing. Specifically, it forwards the traceroute probe based on DA A4::C52 in the IPv6 header. Please note that there is no hop-count expiration at the transit nodes.

- When node N4 receives the packet (B1::0, A4::C52)(A4::C52,A2::C31;

SL=0, HC=62, Flags.O=1; NH=UDP)(Traceroute Probe), it processes the O-bit in SRH, as described in the pseudocode in [I-D.filsfils-spring-srv6-network-programming]. A time-stamped copy of the packet gets punted to the traceroute process for processing. The traceroute process at node N4 checks if its local SID (A2::C31) is locally programmed. If the SID is not locally programmed, it silently drops the packet. Otherwise, it performs the egress check by looking at the SL value in SRH. As SL is equal to zero (i.e., N4 is the egress node), node N4 tries to consume the UDP probe. As UDP probe is set to access an invalid port, the node N4 responses with the ICMPv6 message (Type: Destination unreachable, Code: Port Unreachable).

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

```
> traceroute srv6 A4::C52 via segment-list A2::C31

Tracing the route to SID function A4::C52

 1  2001:DB8:1:2::21 0.512 msec 0.425 msec 0.374 msec
    SRH: (A4::C52, A4::OTP, A2::C31; SL=2)

 2  2001:DB8:3:4::41 0.921 msec 0.816 msec 0.759 msec
    SRH: (A4::C52, A4::OTP, A2::C31; SL=1)
```

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

4. In-situ OAM Applicability

[I-D.brockners-inband-oam-requirements] describes motivation and requirements for In-situ OAM (iOAM). iOAM records operational and telemetry information in the data packet while the packet traverses the network of telemetry domain. iOAM complements out-of-band probe based OAM mechanisms such ICMP ping and traceroute by directly encoding tracing and the other kind of telemetry information to the regular data traffic.

[I-D.brockners-inband-oam-transport] describes transport mechanisms for iOAM data including IPv6 and Segment Routing traffic. Furthermore, [I-D.brockners-inband-oam-data] defines information encoding for iOAM data.

One of the application of iOAM is to perform inband traceroute. In SRv6 network, iOAM traceroute feature can be used to trace the order set of segment ID executed by SRv6 nodes for packet forwarding along

the packet path. This is achieved by recording the node details that the packet traversed in the packet header itself.

Another important application of iOAM is to perform delay measurement in anycast server scenarios. Anycast server deployment is commonly seen for redundancy and load balancing purpose. In SRv6 network, iOAM can be used to collect the timestamp from different anycats servers to measure the delay induced by each server within the anycast cluster that helps to provide SLA constrained services.

One of the other applications of iOAM is to provide the Proof of Transit (POT). Among other features of iOAM, SRv6 networks can use the POT feature of iOAM to verify that all the function SIDs in SRH have been executed before the packet is delivered to the destination. It can also ensure that the order of execution of the SID function has been consistent with the SRH contents.

More details on various applications of iOAM in SRv6 networks will be included in future versions of this document.

5. Seamless BFD Applicability

[RFC7880] defines Seamless BFD (S-BFD) architecture that simplifies BFD mechanism and enables it to perform path monitoring in a controlled and scalable manner. [RFC7881] describes the procedure to perform continuity check using S-BFD in different environments including IPv6 networks. Section 5.1 of [RFC7881] explains the SBFDInitiator specification and procedure to initiate S-BFD control packet in IP and MPLS network. The specification described for IP-routed S-BFD control packet is also directly applicable to the SRv6 network.

S-BFD has a fast bootstrapping capability. Furthermore, in S-BFD, only the ingress is required to keep BFD states; the egress and transit node does not have any knowledge of the BFD session. These attributes of S-BFD make it an excellent candidate for rapid failure detection in the SRv6 network. More details on various S-BFD usage on the SRv6 network will be included in a future version.

6. Monitoring of SRv6 Paths

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

The SR technology enables a centralized OAM entity to perform path monitoring without control plane intervention on monitored nodes.

[I-D.ietf-spring-oam-usecase] describes such centralized OAM mechanism. Specifically, it describes a procedure that can be used to perform path continuity check between any nodes within an SR domain from a centralized monitoring system, with minimal or no control plane intervention on the nodes. However, the document focuses on SR networks with MPLS data plane. The same concept is also applicable to the SRv6 networks. This document describes how the concept can be used to perform path monitoring in an SRv6 network as follows.

In the reference topology in Figure 1, N100 is the controller implementing an END function A100::. In order to verify a segment list <A2::C31, A4::C52>, N100 generates a probe packet with SRH set to (A100::, A4::C52, A2::C31, SL=2). The controller routes the probe packet towards the first segment, which is A2::C31. N2 performs the standard SRH processing and forwards it over link3 with the DA of IPv6 packet set to A4::C52. N4 also performs the normal SRH processing and forwards it over link10 with the DA of IPv6 packet set to A100::. This makes the probe packet loop back to the controller.

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

7. Security Considerations

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

8. IANA Considerations

This document requests IANA to allocate a new Type for ICMPv6 message for "SRv6 OAM".

9. References

9.1. Normative References

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

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

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

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

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

[RFC7880] C.Pignataro, D.Ward, N.Akiya, M.Bhatia, S.Pallagatti, "Seamless Bidirectional Forwarding Detection (S-BFD)", RFC 7880, July 2016.

[RFC7881] C.Pignataro, D.Ward, N.Akiya, "Seamless Bidirectional Forwarding Detection (S-BFD) for IPv4, IPv6, and MPLS", RFC 7881 July 2016.

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

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

9.2. Informative References

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

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

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

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

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

draft-filsfils-spring-segment-routing-policy, work in progress.

10. Acknowledgments

To be added.

Authors' Addresses

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Abstract

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

Table of Contents

1. Introduction

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

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

2. Conventions Used in This Document

2.1. Abbreviations

ECMP: Equal Cost Multi-Path.

SID: Segment ID.

SL: Segment Left.

SR: Segment Routing.

SRH: Segment Routing Header.

SRv6: Segment Routing with IPv6 Data plane.

TC: Traffic Class.

UCMP: Unequal Cost Multi-Path.

2.2. Terminology and Reference Topology

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

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

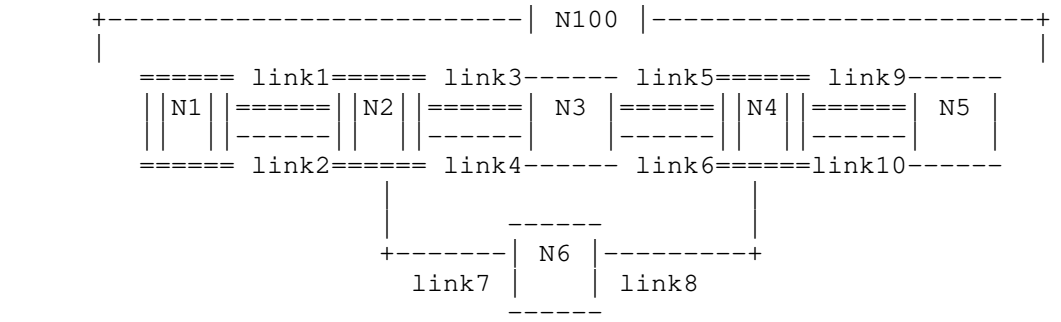


Figure 1 Reference Topology

In the reference topology:

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

Nodes N3, N5 and N6 are classic IPv6 nodes.

Node N100 is a controller.

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

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

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

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

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

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

3. OAM Building Blocks

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

3.1. O-flag in Segment Routing Header

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

3.2. OAM Segments

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

3.2.1. End.OP: OAM Endpoint with Punt

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

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

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

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

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

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

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

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

1. Timestamp the packet ;; Ref1

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

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

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

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

4. OAM Mechanisms

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

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

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

4.1. Ping

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

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

4.1.1. Classic Ping

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

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

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

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

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

Figure 2 A sample ping output at an SRv6 capable node

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

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

4.1.2. Pinging a SID Function

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2. Error Reporting

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

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

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

4.3. Traceroute

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

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

4.3.1. Classic Traceroute

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

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

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

Tracing the route to B5::

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

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

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

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

Figure 3 A sample traceroute output at an SRv6 capable node

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

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

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

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

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

4.3.2. Traceroute to a SID Function

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

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

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

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

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

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

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

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

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

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

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

Figure 4 displays a sample traceroute output for this example.

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

Tracing the route to SID function B:4:C52

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

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

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

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

4.3.2.2. Tracing SRv6 Overlay

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

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

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

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

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

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

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

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

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

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

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

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

4.5. Monitoring of SRv6 Paths

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

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

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

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

5. Security Considerations

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

6. IANA Considerations

6.1. ICMPv6 type Numbers Registry

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

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

TBA (suggested value: 162) SRv6 OAM Message.

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

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

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

6.3. SRv6 OAM Endpoint Types

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

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

7. References

7.1. Normative References

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

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

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

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

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

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

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

7.2. Informative References

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

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

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

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

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

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

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

8. Acknowledgments

To be added.

Authors' Addresses

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 7, 2018

D. Dukes, Ed.
C. Filsfils
Cisco Systems
G. Dawra
LinkedIn
X. Xu
Alibaba
D. Voyer
Bell Canada
P. Camarillo
F. Clad
Cisco Systems
S. Salsano
Univ. of Rome Tor Vergata
June 5, 2018

SR For SDWAN: VPN with Underlay SLA
draft-dukes-spring-sr-for-sdwan-00

Abstract

This document describes how SR enables underlay Service Level Agreements (SLA) to a VPN with scale and security while ensuring service opacity. This solution applies to Over-The-Top VPN (OTT VPN) and Software-Defined WAN (SDWAN).

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 December 7, 2018.

Table of Contents

1. Introduction

This document describes how SR enables underlay SLA to a VPN with scale and security while ensuring service opacity. This solution applies to Over-The-Top VPN (OTT VPN) with SLA differentiation, and Software-Defined WAN (SDWAN) with SLA differentiation.

The body of this text uses SRv6 for illustration. A similar solution leveraging SR-MPLS is illustrated in an appendix.

This document assumes familiarity with the following IETF documents:

- Segment Routing Architecture [I-D.ietf-spring-segment-routing]
- Segment Routing with MPLS data plane [I-D.ietf-spring-segment-routing-mpls]
- IPv6 Segment Routing Header [I-D.ietf-6man-segment-routing-header]
- SRv6 Network Programming [I-D.filsfils-spring-srv6-network-programming]
- Segment Routing Policy For Traffic Engineering [I-D.filsfils-spring-segment-routing-policy]
- IS-IS Extensions to Support Segment Routing over IPv6 Dataplane [I-D.bashandy-isis-srv6-extensions]

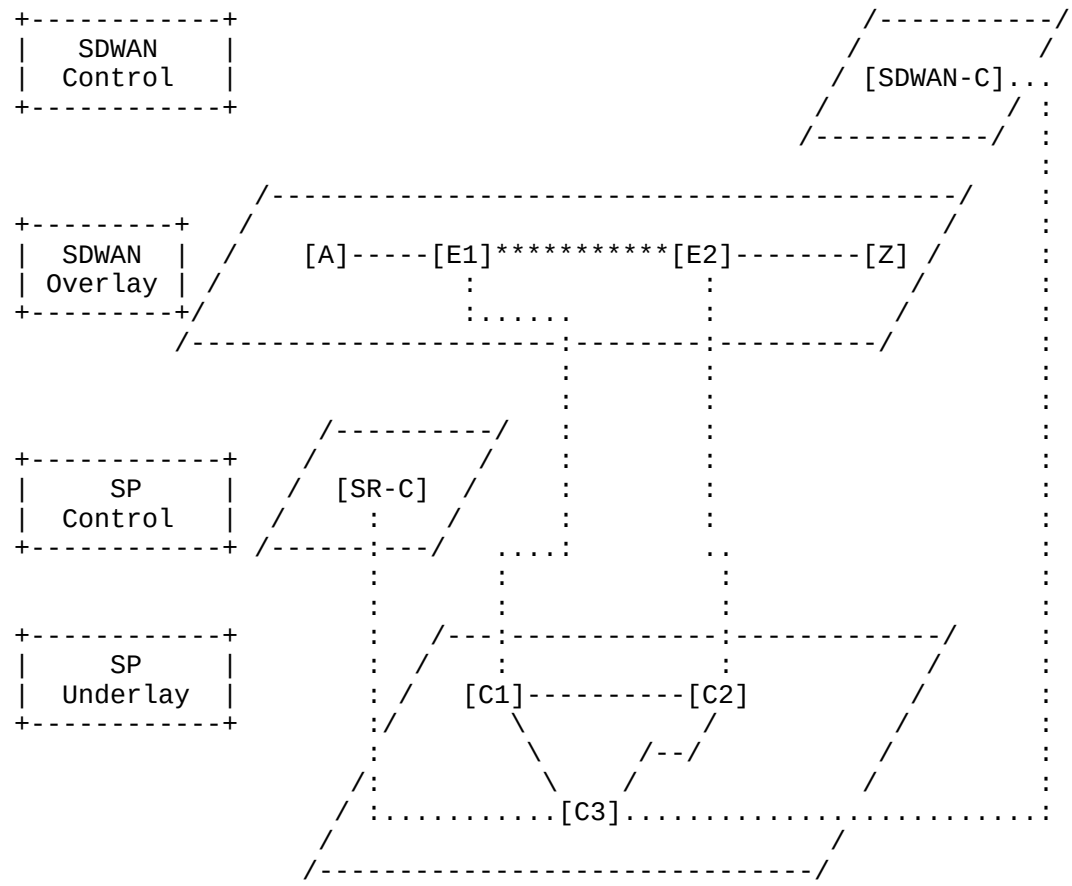
For clarity, this version of the document uses the SDWAN example with SRv6 to illustrate how SR can be used to provide underlay SLA to overlay services. The journey of a packet from the left site to the right site of the SDWAN Overlay is described. The solution applies similarly for the return path.

2. Requirements Notation

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

3. Single Provider

3.1. Directly Connected CE to PE



**** = logical connection
:... = physical connection, between layers
/--\ = physical connection, within a layer

Figure 1: SDWAN Reference Diagram

An SDWAN overlay is composed of two sites A and Z, connected to the Internet via edge nodes E1 and E2 respectively. E1 and E2 (customer edge nodes) are connected via a Service Provider (SP) underlay to form the VPN between the sites.

C1, C2 and C3 are nodes of the SP underlay, where C1 and C2 are Provider Edge nodes. ISIS is deployed in the SP underlay with the same cost on each link.

E1 and E2 connect to C1 and C2 respectively. The shortest path from C1 to C2 is the best-effort path. The explicit path C1-C3-C2 is the

low-latency path. By default, traffic transported from C1 to C2 follows the best-effort path. By default, an SDWAN cannot benefit from the low-latency path from C1 to C2.

The address of A is 10.10.0.10/32 and the address of Z is 10.26.0.26/32. E1 and E2 respectively advertise 10.10/16 and 10.26/16 to the SDWAN controller SDWAN-C via a secure channel over the Internet. The solution is applicable to any traffic exchanged between the sites, including IPv4, IPv6 or L2. For clarity, a single example with IPv4 in the SDWAN Overlay is used.

The SP operates an SR controller SR-C capable of computing constrained paths from C1 to C2.

3.2. Best-effort Underlay Transport

Let’s consider the path taken by traffic from A to Z, across the SDWAN, between nodes E1 and E2 with addresses E1:: and E2:: respectively.

Host A sends a packet P to Z via E1. Packet P has source address 10.10.0.10 and destination address 10.26.0.26, illustrated as P (10.10.0.10,10.26.0.26)(payload). E1, upon receipt of P, determines E2 is the edge node to be used to reach Z. Edge node E1 encrypts, encapsulates and forwards the packet P toward E2 and Z, and it is handled as follow:

- Between A and E1 : P (10.10.0.10,10.26.0.26)(Payload)
- Between E1 and C1 : P (E1::,E2::,NH=ESP)(NH=IPv4,(10.10.0.10,10.26.0.26)(Payload))
  - Note that ESP tunnel mode encapsulation, encryption and authentication is assumed but not required.
- Between C1 and C2 : P (E1::,E2::,NH=ESP)(NH=IPv4,(10.10.0.10,10.26.0.26)(Payload))
- Between C2 and E2 : P (E1::,E2::,NH=ESP)(NH=IPv4,(10.10.0.10,10.26.0.26)(Payload))
- Between E2 and Z : P (10.10.0.10,10.26.0.26)(Payload)

This example illustrates that, classically (i.e., without the SR solution described in this document), the SDWAN cannot leverage the rich infrastructure of the SP to meet its needs. The SP is constrained to offer best-effort transit which does not reflect the capabilities of its infrastructure.

3.3. SR for Underlay SLA Differentiation

SR enables the SDWAN to steer selected flows through selected transport paths of the SP, using the same example in Figure 1.

This small example, with only 3 SP routers, assumes all three support SRv6. As explained in [I-D.filsfils-spring-srv6-network-programming], a typical deployment would only require SRv6 at a few strategic waypoints deployed through the network.

It also assumes ISIS supports the lightweight SRv6 extension described in [I-D.bashandy-isis-srv6-extensions].

The illustration convention from [I-D.filsfils-spring-srv6-network-programming] is used such that:

- SRv6 SID Cj:: is explicitly instantiated at node Cj and bound to the END.PSP function.
- SRv6 SID C1::B21 is a Binding SID (BSID) explicitly instantiated at headend C1 and bound to the SRTE policy <C3::, C2::> toward endpoint C2.
  - Note the return direction would use a BSID C2::B11, bound at headend C2, to the SRTE policy <C3::, C1::> toward endpoint C1.

The Control-Plane (CP) workflow that leads to the instantiation of this Binding SID will be explained in the Control-Plane section.

Let’s again consider the path from A to Z for a packet P, but this time E1 has been configured by SDWAN-C to steer packet P into a preferred low-latency path of the SP bound to the binding SID C1:B21.

- Between A and E1
  - P (10.10.0.10,10.26.0.26)(payload)
- Between E1 and C1
  - P (E1::,C1::B21; NH=SRH)(E2::,C1::B21; SL=1; NH=ESP)(NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

When the Binding SID C1::B21 is processed at C1, the SR TE Policy is selected and the SRH for SID list <C3::,C2::> is inserted into P:

- Between C1 and C3

* P (E1::,C3::;NH=SRH)(E2::,C2::,C3::; SL=2;NH=ESP) (NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

At C3, the SegmentsLeft is decremented as the END SID C3:: is processed, and C2:: is placed in the destination address:

- Between C3 and C2
  * P (E1::,C2::;NH=SRH)(E2::,C2::,C3::; SL=1;NH=ESP) (NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

At C2, the SegmentsLeft is decremented to 0, and penultimate segment pop is applied as the END SID C2:: is processed and E2:: is placed in the destination address while the SRH is removed:

- Between C2 and E2
  * P (E1::,E2::,NH=ESP)(NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

Finally, E2 decrypts the packet and strips the outer header to forward the original packet to Z:

- Between E2 and Z
  * P (10.10.0.10,10.26.0.26)(Payload)

The SDWAN edge nodes (E1,E2) maintain their existing behavior of

- Ingress Edge Node: classify ingress traffic, determining the egress edge node, selecting a local output interface, secure the traffic, and forward to the chosen egress edge node.
- Egress Edge Node: decapsulate, decrypt and forward on the internal network.

The only change is that the Ingress node now monitors and selects an SRv6 binding SID then pushes an SRH with two SIDs.

Note as well that the ingress and egress edge nodes never see the actual SID list used by the SP to deliver the preferred path. A variation of this design allows for the BSID to be kept in the packet so that the egress node can detect which packets have been steered on which preferred path (for accounting or monitoring purposes).

This is a fairly simple example of how SRv6 binding SIDs and SR TE policies may be used to provide multiple diverse paths for SDWAN traffic traversing a single provider network.

3.4. Accounting

As per SRv6 network programming [I-D.filsfils-spring-srv6-network-programming], each SRTE policy and its bound BSID is associated with a unique traffic counter. This allows the SP to implement various forms of billing and reporting to the customer of the preferred path.

3.5. Security

The domain of trust security solution documented in [I-D.filsfils-spring-srv6-network-programming] is utilized.

Specifically SEC1, SEC2 and SEC3 guarantee that external traffic to the SP cannot exercise the SID's of the SP.

The following behavior is added: the ACL implementing SEC1 and SEC2 on node C1 is updated to specifically allow traffic from E1:: to C1::B21.

Only the SDWAN edge that has ordered the preferential service can use it.

Any other customer of the SP is unable to use the preferential path bound to BSID C1::B21.

The SDWAN site that has ordered the preferential service is unable to directly program the network of the SP using the internal SID's of the SP. The SDWAN edge node is restricted to the BSID, which opacifies the SP operation.

3.6. Remotely Connected (to PE)

Well known authentication technology with details provided in subsequent revisions will be added, detailing the scenario with SDWAN edge nodes not directly connected to the SP node terminating the binding SID.

4. Multiple Providers

Well known authentication technology with details provided in subsequent revisions will be added, detailing the scenario with SDWAN edge nodes connected to the SP node offering binding SID via an intermediate SP.

5. Control Plane

The SDWAN overlay in Figure 1 is managed by an SDWAN controller, SDWAN-C.

The control protocols used by the SDWAN-C to signal the site routes, the BSID’s and the site policies (which traffic on which BSID when) securely over the SP network to E1 and E2 is outside the scope of this document.

The SP underlay operates its internal SR deployment with an SR controller (SR-C). SR-C interacts with the SP’s network (Cj) through standardized protocols (PCE[RFC4674] , PCEP [RFC5440]/[RFC4657], BGP RR[RFC4456], BGP-TE [I-D.ietf-idr-segment-routing-te-policy], BGP-LS [RFC7752])

Most likely, the SP would operate its underlay SLA service with a service controller (SERV-C) that is separate from SR-C. To simplify the illustration, this text assumes that SERV-C and SR-C are integrated.

This section describes the high-level interaction between these controllers for the low-latency use-case described in this document, where an enterprise operator installs a policy in the SDWAN-C requiring a low latency service between E1 and E2.

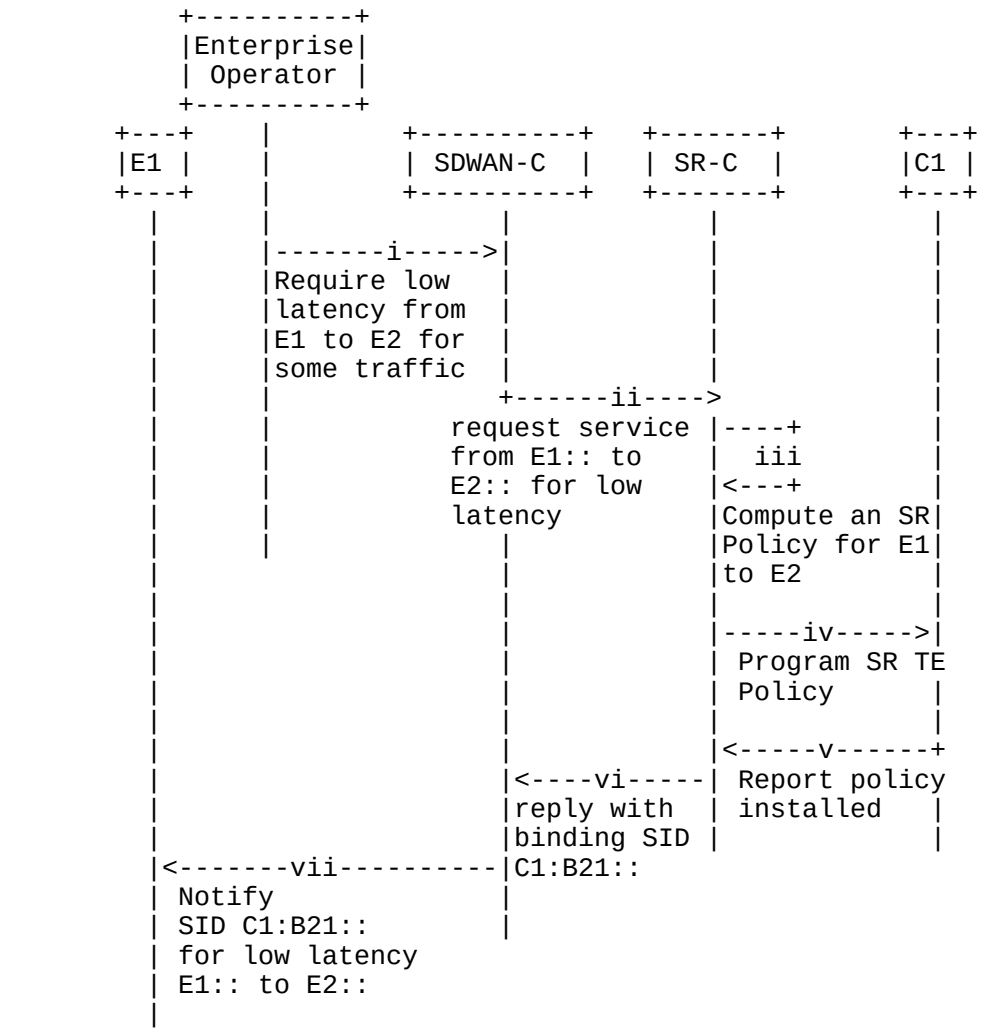


Figure 2: Controlplane Flow

(i) The enterprise operator requests a low-latency path from site E1 to site E2. It defines which traffic needs to be steered on this preferred path.

(ii) SDWAN-C requests a low-latency service from SR-C for the public address of E1 to the public address of E2.

(iii) SR-C computes an SR Policy to satisfy SDWAN-C’s request:

A. SR-C maps the E1 and E2 addresses to its managed nodes C1 and C2.

B. SR-C statefully registers the SRTE policy from C1 to C2 for low-latency.

C. SR-C computes the SID list fulfilling the SLA requirement (e.g. <C3::, C2::>). The stateful nature of the SRTE policy ensures that the SID list is updated whenever required due to network state change.

D. SR-C binds a stable Binding SID C1::B21 to the SRTE policy.

(iv) SR-C programs C1 with the computed SRTE policy and the selected BSID. Standardized protocols such as [I-D.ietf-idr-segment-routing-te-policy] or [RFC5440] are used.

(v) C1 installs the policy in its dataplane and reports the status of the SRTE policy to SR-C using standardized protocols [RFC7752] or [RFC5440] and [I-D.negi-pce-segment-routing-ipv6].

(vi) SR-C replies to SDWAN-C with BSID C1::B21

(vii) SDWAN-C programs E1 with the flow classification and steering policy to insert SRv6 SID C1::B21 on the appropriate traffic

6. Benefits

6.1. Scale

The SP network does not hold any per-SDWAN-flow state in the core of its network.

The SP network does not hold any complex L3-L7 flow classification at the edge of its network.

The SP network is unaware of any policy change of the SDWAN instance either in terms of which flow to classify, when to steer it and on which path.

The SP’s role only consists in statefully maintaining SRTE policies at the edge of the network and maintaining a few 100’s of SID’s inside its core network. This is the stateless property of Segment Routing.

6.2. Privacy

The SP network does not share any information of its infrastructure, topology, capacity, internal SID's.

The SDWAN instance does not share any information on its traffic classification, steering policy and business logic.

6.3. Flexible Billing

The traffic destined to a BSID is individually accounted [I-D.filsfils-spring-srv6-network-programming].

The SP and SDWAN instance can agree on various forms of billing for the usage of the preferential path.

6.4. Security

By default, the SP's SR infrastructure is protected by the simple domain of trust solution documented in [I-D.filsfils-spring-srv6-network-programming].

A BSID (and the related preferential path) can only be accessed by the specific SDWAN instance (and site) that ordered the service.

The security solution supports any SDWAN site connection type: directly connected to the SP edge or not.

7. Appendix

7.1. Single Provider Example Using End.BM With an MPLS Core

Reusing the example from Section 3.3, with an SP core that supports SR MPLS as defined in [I-D.ietf-spring-segment-routing-mpls]. Each node C1, C2 and C3 have node-SIDs defined, resulting in labels 16001, 16002, and 16003 respectively. In such a case a packet from A to Z has an SRv6 binding SID applied, associated with an SR policy at node C1. Node C1 translates the binding SID to an MPLS label stack which is pushed on the packet.

For example:

- SRv6 SID C1::B22 is a Binding SID (BSID) explicitly instantiated at headend C1 and bound to the SRTE policy <16003,16002> toward endpoint C2.

    * Note the return direction would use a BSID C2::B12, bound at headend C2, to the SRTE policy <16003, 16001> toward endpoint C1.

Let’s again consider the path from A to Z for a packet P where E1 has been configured by SDWAN-C to steer packet P into a preferred low-latency path of the SP bound to the binding SID C1::B22.

- Between A and E1
    * P (10.10.0.10,10.26.0.26)(payload)
- Between E1 and C1
    * P (E1::,C1::B22; NH=SRH)(E2::,C1::B22; SL=1; NH=ESP)(NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

When the Binding SID C1::B22 is processed at C1, the SR TE Policy is selected and the label stack for SID list <16003,16002> is pushed on P. In practice 16003 is not pushed on the wire since it has been distributed with PHP:

- Between C1 and C3
    * P (16002)(E1::,E2::;NH=ESP) (NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

At C3, 16002 is popped and PHP requires no new label be pushed as P is forwarded via the link to C2:

- Between C3 and C2
    * P (E1::,E2::;NH=ESP)(NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

At C2, there is no more label stack so it forwards E2:: using the global IPv6 table toward E2:

- Between C2 and E2
    * P (E1::,E2::,NH=ESP)(NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

Finally, E2 decrypts the packet and strips the outer header to forward the original packet to Z:

- Between E2 and Z
    * P (10.10.0.10,10.26.0.26)(Payload)

Again the only change is that the Ingress node now selects an MPLS binding SID for traffic taking the lowest latency path. The Ingress node has no knowledge of the SP underlay.

7.1.1. Accounting

As defined in Section 3.4

7.1.2. Security

The SRv6 SID is secured as defined in Section 3.5

MPLS is not enabled on the CE-to-PE link.

7.2. Single Provider Example Using MPLS From CE to PE for BSID

To be completed in future revisions

7.3. Single Provider Example Using SRMPLS Over UDP For CE to PE Not Directly Connected Over Internet

Reusing the example from Section 3.3, with an SP core that supports the SR MPLS extensions defined in [I-D.ietf-isis-segment-routing-extensions]. Each node C1, C2 and C3 have node-SIDs defined, resulting in labels 16001, 16002, and 16003 respectively.

Let’s again consider the path from A to Z for a packet P, but this time E1 has been configured by SDWAN-C to steer packet P into a preferred low-latency path of the SP bound to an MPLS binding SID.

For example:

- MPLS label 24102 is a Binding SID (BSID) explicitly instantiated at headend C1 and bound to the SRTE policy <16003,16002> toward endpoint C2.
  - Note the return direction would use a BSID 24201, bound at headend C2, to the SRTE policy <16003, 16001> toward endpoint C1.

Let’s again consider the path from A to Z for a packet P where E1 has been configured by SDWAN-C to steer packet P into a preferred low-latency path of the SP bound to the binding SID 24102.

- Between A and E1
  - P (10.10.0.10,10.26.0.26)(payload)

- Between E1 and C1, RFC7510 UDP encapsulation of MPLS is used to transport the MPLS labeled traffic.
  - P (E1::,C1::; NH=UDP)(24102,(10.10.0.10,10.26.0.26)(Payload))

When C1 receives P, it decapsulates the UDP header and pops the Binding SID 24102, the SR TE Policy is selected and the label stack for SID list <16003,16002> is pushed on P. In practice, 16003 is not pushed on the wire since it has been distributed with PHP.

The remainder of this use case is identical to Section 7.1. The only change is that the Ingress node now uses an MPLS binding SID transported over UDP instead of an SRv6 binding SID, allowing the use of an IPv6 or IPv4 transport from CE to PE.

7.3.1. Accounting

As defined in Section 3.4

7.3.2. Security

[RFC7510] defines the use of DTLS to authenticate and encrypt the MPLS in UDP encapsulation between CE and PE. The authentication ensures the source is authorized to send traffic to a binding SID.

After the source is verified as authorized, the source address and Binding SID SHOULD be checked to determine if the source is permitted to use the specific Binding SID in the MPLS label.

MPLS is not enabled on the CE-to-PE link.

8. IANA Considerations

No current considerations.

9. Security Considerations

A domain of trust is secured via methods documented in [I-D.filsfils-spring-srv6-network-programming]

10. References

10.1. Informative References

[I-D.bashandy-isis-srv6-extensions]
Ginsberg, L., Bashandy, A., Filsfils, C., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Routing over IPv6 Dataplane", draft-bashandy-isis-srv6-extensions-02 (work in progress), March 2018.

[I-D.filsfils-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Hegde, S., daniel.voyer@bell.ca, d., Lin, S., bogdanov@google.com, b., Krol, P., Horneffer, M., Steinberg, D., Decraene, B., Litkowski, S., Mattes, P., Ali, Z., Talaulikar, K., Liste, J., Clad, F., and K. Raza, "Segment Routing Policy Architecture", draft-filsfils-spring-segment-routing-policy-06 (work in progress), May 2018.

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-13 (work in progress), May 2018.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Jain, D., Mattes, P., Rosen, E., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-03 (work in progress), May 2018.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-16 (work in progress), April 2018.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[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-13 (work in progress), April 2018.

[I-D.negi-pce-segment-routing-ipv6]
Negi, M., Kaladharan, P., Dhody, D., and S. Sivabalan, "PCEP Extensions for Segment Routing leveraging the IPv6 data plane", draft-negi-pce-segment-routing-ipv6-01 (work in progress), March 2018.

[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>.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <https://www.rfc-editor.org/info/rfc4456>.

[RFC4657] Ash, J., Ed. and J. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol Generic Requirements", RFC 4657, DOI 10.17487/RFC4657, September 2006, <https://www.rfc-editor.org/info/rfc4657>.

[RFC4674] Le Roux, J., Ed., "Requirements for Path Computation Element (PCE) Discovery", RFC 4674, DOI 10.17487/RFC4674, October 2006, <https://www.rfc-editor.org/info/rfc4674>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <https://www.rfc-editor.org/info/rfc5440>.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <https://www.rfc-editor.org/info/rfc7510>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

10.2. Normative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Li, Z., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., and M. Sharif, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-04 (work in progress), March 2018.

Authors' Addresses

Darren Dukes (editor)
Cisco Systems
Canada

Email: ddukes@cisco.com

Clarence Filsfils
Cisco Systems
Belgium

Email: cfilsfil@cisco.com

Gaurav Dawra
LinkedIn
USA

Email: gdawra@linkedin.com

Xiaohu Xu
Alibaba
China

Email: xiaohu.xxh@alibaba-inc.com

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Pablo Camarillo Garvia
Cisco Systems
Spain

Email: pcamaril@cisco.com

Francois Clad
Cisco Systems
France

Stefano Salsano
Univ. of Rome Tor Vergata
Italy

Email: stefano.salsano@uniroma2.it

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 7, 2019

D. Dukes, Ed.
C. Filsfils
Cisco Systems
G. Dawra
LinkedIn
X. Xu
Alibaba
D. Voyer
Bell Canada
P. Camarillo
F. Clad
Cisco Systems
S. Salsano
Univ. of Rome Tor Vergata

December 4, 2018

SR For SDWAN: VPN with Underlay SLA
draft-dukes-spring-sr-for-sdwan-01

Abstract

This document describes how SR enables underlay Service Level Agreements (SLA) to a VPN with scale and security while ensuring service opacity. This solution applies to Over-The-Top VPN (OTT VPN) and Software-Defined WAN (SDWAN).

Table of Contents

1. Introduction

This document describes how SR enables underlay SLA to a VPN with scale and security while ensuring service opacity. This solution applies to Over-The-Top VPN (OTT VPN) with SLA differentiation, and Software-Defined WAN (SDWAN) with SLA differentiation.

The body of this text uses SRv6 for illustration. A similar solution leveraging SR-MPLS is illustrated in an appendix.

This document assumes familiarity with the following IETF documents:

- Segment Routing Architecture [I-D.ietf-spring-segment-routing]
- Segment Routing with MPLS data plane [I-D.ietf-spring-segment-routing-mpls]
- IPv6 Segment Routing Header [I-D.ietf-6man-segment-routing-header]
- SRv6 Network Programming [I-D.filsfils-spring-srv6-network-programming]
- Segment Routing Policy For Traffic Engineering [I-D.filsfils-spring-segment-routing-policy]
- IS-IS Extensions to Support Segment Routing over IPv6 Dataplane [I-D.bashandy-isis-srv6-extensions]

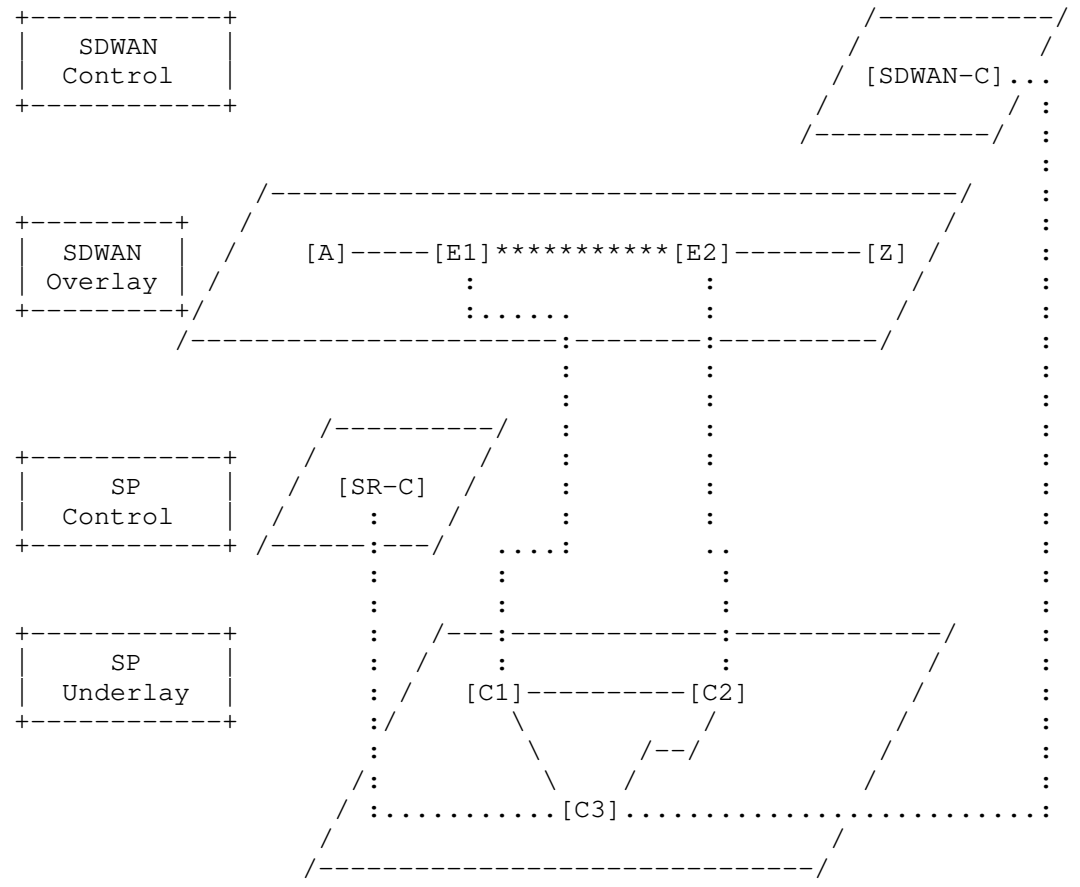
For clarity, this version of the document uses the SDWAN example with SRv6 to illustrate how SR can be used to provide underlay SLA to overlay services. The journey of a packet from the left site to the right site of the SDWAN Overlay is described. The solution applies similarly for the return path.

2. Requirements Notation

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

3. Single Provider

3.1. Directly Connected CE to PE



**** = logical connection
:... = physical connection, between layers
/--\ = physical connection, within a layer

Figure 1: SDWAN Reference Diagram

An SDWAN overlay is composed of two sites A and Z, connected to the Internet via edge nodes E1 and E2 respectively. E1 and E2 (customer edge nodes) are connected via a Service Provider (SP) underlay to form the VPN between the sites.

C1, C2 and C3 are nodes of the SP underlay, where C1 and C2 are Provider Edge nodes. ISIS is deployed in the SP underlay with the same cost on each link.

E1 and E2 connect to C1 and C2 respectively. The shortest path from C1 to C2 is the best-effort path. The explicit path C1-C3-C2 is the

low-latency path. By default, traffic transported from C1 to C2 follows the best-effort path. By default, an SDWAN cannot benefit from the low-latency path from C1 to C2.

The address of A is 10.10.0.10/32 and the address of Z is 10.26.0.26/32. E1 and E2 respectively advertise 10.10/16 and 10.26/16 to the SDWAN controller SDWAN-C via a secure channel over the Internet. The solution is applicable to any traffic exchanged between the sites, including IPv4, IPv6 or L2. For clarity, a single example with IPv4 in the SDWAN Overlay is used.

The SP operates an SR controller SR-C capable of computing constrained paths from C1 to C2.

3.2. Best-effort Underlay Transport

Let's consider the path taken by traffic from A to Z, across the SDWAN, between nodes E1 and E2 with addresses E1:: and E2:: respectively.

Host A sends a packet P to Z via E1. Packet P has source address 10.10.0.10 and destination address 10.26.0.26, illustrated as P (10.10.0.10,10.26.0.26)(payload). E1, upon receipt of P, determines E2 is the edge node to be used to reach Z. Edge node E1 encrypts, encapsulates and forwards the packet P toward E2 and Z, and it is handled as follow:

- Between A and E1 : P (10.10.0.10,10.26.0.26)(Payload)
- Between E1 and C1 : P (E1::,E2::,NH=ESP)(NH=IPv4,(10.10.0.10,10.26.0.26)(Payload))
  - Note that ESP tunnel mode encapsulation, encryption and authentication is assumed but not required.
- Between C1 and C2 : P (E1::,E2::,NH=ESP)(NH=IPv4,(10.10.0.10,10.26.0.26)(Payload))
- Between C2 and E2 : P (E1::,E2::,NH=ESP)(NH=IPv4,(10.10.0.10,10.26.0.26)(Payload))
- Between E2 and Z : P (10.10.0.10,10.26.0.26)(Payload)

This example illustrates that, classically (i.e., without the SR solution described in this document), the SDWAN cannot leverage the rich infrastructure of the SP to meet its needs. The SP is constrained to offer best-effort transit which does not reflect the capabilities of its infrastructure.

3.3. SR for Underlay SLA Differentiation

SR enables the SDWAN to steer selected flows through selected transport paths of the SP, using the same example in Figure 1.

This small example, with only 3 SP routers, assumes all three support SRv6. As explained in [I-D.filsfils-spring-srv6-network-programming], a typical deployment would only require SRv6 at a few strategic waypoints deployed through the network.

It also assumes ISIS supports the lightweight SRv6 extension described in [I-D.bashandy-isis-srv6-extensions].

The illustration convention from [I-D.filsfils-spring-srv6-network-programming] is used such that:

- SRv6 SID Cj:: is explicitly instantiated at node Cj and bound to the END.PSP function.
- SRv6 SID C1::B21 is a Binding SID (BSID) explicitly instantiated at headend C1 and bound to the SRTE policy <C3::, C2::> toward endpoint C2.
  - Note the return direction would use a BSID C2::B11, bound at headend C2, to the SRTE policy <C3::, C1::> toward endpoint C1.

The Control-Plane (CP) workflow that leads to the instantiation of this Binding SID will be explained in the Control-Plane section.

Let's again consider the path from A to Z for a packet P, but this time E1 has been configured by SDWAN-C to steer packet P into a preferred low-latency path of the SP bound to the binding SID C1:B21.

- Between A and E1
  - P (10.10.0.10,10.26.0.26)(payload)
- Between E1 and C1
  - P (E1::,C1::B21; NH=SRH)(E2::,C1::B21; SL=1; NH=ESP)(NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

When the Binding SID C1::B21 is processed at C1, the SR TE Policy is selected and the SRH for SID list <C3::,C2::> is inserted into P:

- Between C1 and C3

* P (E1::,C3::;NH=SRH)(E2::,C2::,C3::; SL=2;NH=ESP) (NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

At C3, the SegmentsLeft is decremented as the END SID C3:: is processed, and C2:: is placed in the destination address:

- Between C3 and C2
  * P (E1::,C2::;NH=SRH)(E2::,C2::,C3::; SL=1;NH=ESP) (NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

At C2, the SegmentsLeft is decremented to 0, and penultimate segment pop is applied as the END SID C2:: is processed and E2:: is placed in the destination address while the SRH is removed:

- Between C2 and E2
  * P (E1::,E2::,NH=ESP)(NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

Finally, E2 decrypts the packet and strips the outer header to forward the original packet to Z:

- Between E2 and Z
  * P (10.10.0.10,10.26.0.26)(Payload)

The SDWAN edge nodes (E1,E2) maintain their existing behavior of

- Ingress Edge Node: classify ingress traffic, determining the egress edge node, selecting a local output interface, secure the traffic, and forward to the chosen egress edge node.
- Egress Edge Node: decapsulate, decrypt and forward on the internal network.

The only change is that the Ingress node now monitors and selects an SRv6 binding SID then pushes an SRH with two SIDs.

Note as well that the ingress and egress edge nodes never see the actual SID list used by the SP to deliver the preferred path. A variation of this design allows for the BSID to be kept in the packet so that the egress node can detect which packets have been steered on which preferred path (for accounting or monitoring purposes).

This is a fairly simple example of how SRv6 binding SIDs and SR TE policies may be used to provide multiple diverse paths for SDWAN traffic traversing a single provider network.

3.4. Accounting

As per SRv6 network programming [I-D.filsfils-spring-srv6-network-programming], each SRTE policy and its bound BSID is associated with a unique traffic counter. This allows the SP to implement various forms of billing and reporting to the customer of the preferred path.

3.5. Security

The domain of trust security solution documented in [I-D.filsfils-spring-srv6-network-programming] is utilized.

Specifically SEC1, SEC2 and SEC3 guarantee that external traffic to the SP cannot exercise the SID's of the SP.

The following behavior is added: the ACL implementing SEC1 and SEC2 on node C1 is updated to specifically allow traffic from E1:: to C1::B21.

Only the SDWAN edge that has ordered the preferential service can use it.

Any other customer of the SP is unable to use the preferential path bound to BSID C1::B21.

The SDWAN site that has ordered the preferential service is unable to directly program the network of the SP using the internal SID's of the SP. The SDWAN edge node is restricted to the BSID, which opacifies the SP operation.

3.6. Remotely Connected (to PE)

Well known authentication technology with details provided in subsequent revisions will be added, detailing the scenario with SDWAN edge nodes not directly connected to the SP node terminating the binding SID.

4. Multiple Providers

Well known authentication technology with details provided in subsequent revisions will be added, detailing the scenario with SDWAN edge nodes connected to the SP node offering binding SID via an intermediate SP.

5. Control Plane

The SDWAN overlay in Figure 1 is managed by an SDWAN controller, SDWAN-C.

The control protocols used by the SDWAN-C to signal the site routes, the BSID's and the site policies (which traffic on which BSID when) securely over the SP network to E1 and E2 is outside the scope of this document.

The SP underlay operates its internal SR deployment with an SR controller (SR-C). SR-C interacts with the SP's network (Cj) through standardized protocols (PCE[RFC4674] , PCEP [RFC5440]/[RFC4657], BGP RR[RFC4456], BGP-TE [I-D.ietf-idr-segment-routing-te-policy], BGP-LS [RFC7752])

Most likely, the SP would operate its underlay SLA service with a service controller (SERV-C) that is separate from SR-C. To simplify the illustration, this text assumes that SERV-C and SR-C are integrated.

This section describes the high-level interaction between these controllers for the low-latency use-case described in this document, where an enterprise operator installs a policy in the SDWAN-C requiring a low latency service between E1 and E2.

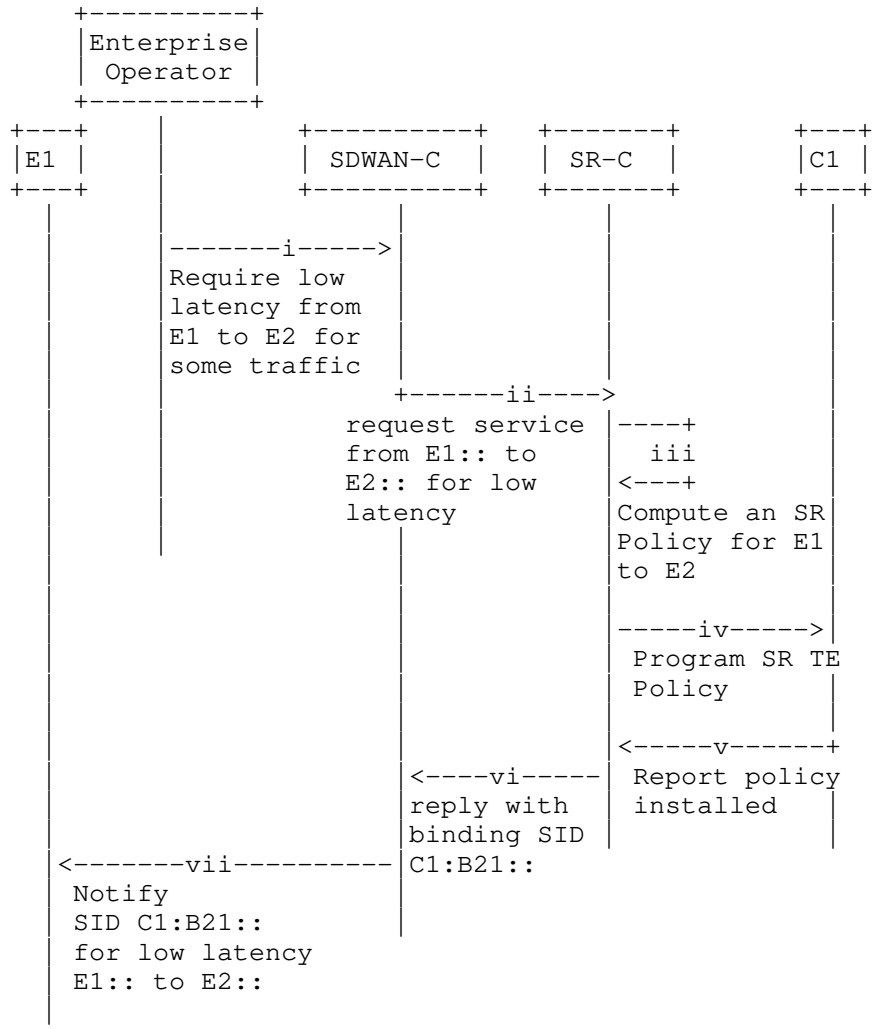


Figure 2: Controlplane Flow

(i) The enterprise operator requests a low-latency path from site E1 to site E2. It defines which traffic needs to be steered on this preferred path.

(ii) SDWAN-C requests a low-latency service from SR-C for the public address of E1 to the public address of E2.

(iii) SR-C computes an SR Policy to satisfy SDWAN-C's request:

A. SR-C maps the E1 and E2 addresses to its managed nodes C1 and C2.

B. SR-C statefully registers the SRTE policy from C1 to C2 for low-latency.

C. SR-C computes the SID list fulfilling the SLA requirement (e.g. <C3::, C2::>). The stateful nature of the SRTE policy ensures that the SID list is updated whenever required due to network state change.

D. SR-C binds a stable Binding SID C1::B21 to the SRTE policy.

(iv) SR-C programs C1 with the computed SRTE policy and the selected BSID. Standardized protocols such as [I-D.ietf-idr-segment-routing-te-policy] or [RFC5440] are used.

(v) C1 installs the policy in its dataplane and reports the status of the SRTE policy to SR-C using standardized protocols [RFC7752] or [RFC5440] and [I-D.negi-pce-segment-routing-ipv6].

(vi) SR-C replies to SDWAN-C with BSID C1::B21

(vii) SDWAN-C programs E1 with the flow classification and steering policy to insert SRv6 SID C1::B21 on the appropriate traffic

6. Benefits

6.1. Scale

The SP network does not hold any per-SDWAN-flow state in the core of its network.

The SP network does not hold any complex L3-L7 flow classification at the edge of its network.

The SP network is unaware of any policy change of the SDWAN instance either in terms of which flow to classify, when to steer it and on which path.

The SP's role only consists in statefully maintaining SRTE policies at the edge of the network and maintaining a few 100's of SID's inside its core network. This is the stateless property of Segment Routing.

6.2. Privacy

The SP network does not share any information of its infrastructure, topology, capacity, internal SID's.

The SDWAN instance does not share any information on its traffic classification, steering policy and business logic.

6.3. Flexible Billing

The traffic destined to a BSID is individually accounted [I-D.filsfils-spring-srv6-network-programming].

The SP and SDWAN instance can agree on various forms of billing for the usage of the preferential path.

6.4. Security

By default, the SP's SR infrastructure is protected by the simple domain of trust solution documented in [I-D.filsfils-spring-srv6-network-programming].

A BSID (and the related preferential path) can only be accessed by the specific SDWAN instance (and site) that ordered the service.

The security solution supports any SDWAN site connection type: directly connected to the SP edge or not.

7. Appendix

7.1. Single Provider Example Using End.BM With an MPLS Core

Reusing the example from Section 3.3, with an SP core that supports SR MPLS as defined in [I-D.ietf-spring-segment-routing-mpls]. Each node C1, C2 and C3 have node-SIDs defined, resulting in labels 16001, 16002, and 16003 respectively. In such a case a packet from A to Z has an SRv6 binding SID applied, associated with an SR policy at node C1. Node C1 translates the binding SID to an MPLS label stack which is pushed on the packet.

For example:

- SRv6 SID C1::B22 is a Binding SID (BSID) explicitly instantiated at headend C1 and bound to the SRTE policy <16003,16002> toward endpoint C2.

  * Note the return direction would use a BSID C2::B12, bound at headend C2, to the SRTE policy <16003, 16001> toward endpoint C1.

Let's again consider the path from A to Z for a packet P where E1 has been configured by SDWAN-C to steer packet P into a preferred low-latency path of the SP bound to the binding SID C1::B22.

- Between A and E1
  * P (10.10.0.10,10.26.0.26)(payload)
- Between E1 and C1
  * P (E1::,C1::B22; NH=SRH)(E2::,C1::B22; SL=1; NH=ESP)(NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

When the Binding SID C1::B22 is processed at C1, the SR TE Policy is selected and the label stack for SID list <16003,16002> is pushed on P. In practice 16003 is not pushed on the wire since it has been distributed with PHP:

- Between C1 and C3
  * P (16002)(E1::,E2::;NH=ESP) (NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

At C3, 16002 is popped and PHP requires no new label be pushed as P is forwarded via the link to C2:

- Between C3 and C2
  * P (E1::,E2::;NH=ESP)(NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

At C2, there is no more label stack so it forwards E2:: using the global IPv6 table toward E2:

- Between C2 and E2
  * P (E1::,E2::,NH=ESP)(NH=IPv4(10.10.0.10,10.26.0.26)(Payload))

Finally, E2 decrypts the packet and strips the outer header to forward the original packet to Z:

- Between E2 and Z
  * P (10.10.0.10,10.26.0.26)(Payload)

Again the only change is that the Ingress node now selects an MPLS binding SID for traffic taking the lowest latency path. The Ingress node has no knowledge of the SP underlay.

7.1.1. Accounting

As defined in Section 3.4

7.1.2. Security

The SRv6 SID is secured as defined in Section 3.5

MPLS is not enabled on the CE-to-PE link.

7.2. Single Provider Example Using MPLS From CE to PE for BSID

To be completed in future revisions

7.3. Single Provider Example Using SRMPLS Over UDP For CE to PE Not Directly Connected Over Internet

Reusing the example from Section 3.3, with an SP core that supports the SR MPLS extensions defined in [I-D.ietf-isis-segment-routing-extensions]. Each node C1, C2 and C3 have node-SIDs defined, resulting in labels 16001, 16002, and 16003 respectively.

Let's again consider the path from A to Z for a packet P, but this time E1 has been configured by SDWAN-C to steer packet P into a preferred low-latency path of the SP bound to an MPLS binding SID.

For example:

- MPLS label 24102 is a Binding SID (BSID) explicitly instantiated at headend C1 and bound to the SRTE policy <16003,16002> toward endpoint C2.
  - Note the return direction would use a BSID 24201, bound at headend C2, to the SRTE policy <16003, 16001> toward endpoint C1.

Let's again consider the path from A to Z for a packet P where E1 has been configured by SDWAN-C to steer packet P into a preferred low-latency path of the SP bound to the binding SID 24102.

- Between A and E1
  - P (10.10.0.10,10.26.0.26)(payload)

- Between E1 and C1, RFC7510 UDP encapsulation of MPLS is used to transport the MPLS labeled traffic.

  * P (E1::,C1::; NH=UDP)(24102,(10.10.0.10,10.26.0.26)(Payload))

When C1 receives P, it decapsulates the UDP header and pops the Binding SID 24102, the SR TE Policy is selected and the label stack for SID list <16003,16002> is pushed on P. In practice, 16003 is not pushed on the wire since it has been distributed with PHP.

The remainder of this use case is identical to Section 7.1. The only change is that the Ingress node now uses an MPLS binding SID transported over UDP instead of an SRv6 binding SID, allowing the use of an IPv6 or IPv4 transport from CE to PE.

7.3.1. Accounting

As defined in Section 3.4

7.3.2. Security

[RFC7510] defines the use of DTLS to authenticate and encrypt the MPLS in UDP encapsulation between CE and PE. The authentication ensures the source is authorized to send traffic to a binding SID.

After the source is verified as authorized, the source address and Binding SID SHOULD be checked to determine if the source is permitted to use the specific Binding SID in the MPLS label.

MPLS is not enabled on the CE-to-PE link.

8. IANA Considerations

No current considerations.

9. Security Considerations

A domain of trust is secured via methods documented in [I-D.filsfils-spring-srv6-network-programming]

10. References

10.1. Informative References

[I-D.bashandy-isis-srv6-extensions]
Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Routing over IPv6 Dataplane", draft-bashandy-isis-srv6-extensions-04 (work in progress), October 2018.

[I-D.filsfils-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Hegde, S., daniel.voyer@bell.ca, d., Lin, S., bogdanov@google.com, b., Krol, P., Horneffer, M., Steinberg, D., Decraene, B., Litkowski, S., Mattes, P., Ali, Z., Talaulikar, K., Liste, J., Clad, F., and K. Raza, "Segment Routing Policy Architecture", draft-filsfils-spring-segment-routing-policy-06 (work in progress), May 2018.

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-15 (work in progress), October 2018.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Jain, D., Mattes, P., Rosen, E., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-05 (work in progress), November 2018.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-21 (work in progress), December 2018.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[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-17 (work in progress), December 2018.

[I-D.negi-pce-segment-routing-ipv6]
Negi, M., Dhody, D., Sivabalan, S., and P. Kaladharan, "PCEP Extensions for Segment Routing leveraging the IPv6 data plane", draft-negi-pce-segment-routing-ipv6-03 (work in progress), October 2018.

[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>.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <https://www.rfc-editor.org/info/rfc4456>.

[RFC4657] Ash, J., Ed. and J. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol Generic Requirements", RFC 4657, DOI 10.17487/RFC4657, September 2006, <https://www.rfc-editor.org/info/rfc4657>.

[RFC4674] Le Roux, J., Ed., "Requirements for Path Computation Element (PCE) Discovery", RFC 4674, DOI 10.17487/RFC4674, October 2006, <https://www.rfc-editor.org/info/rfc4674>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <https://www.rfc-editor.org/info/rfc5440>.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <https://www.rfc-editor.org/info/rfc7510>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

10.2. Normative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-06 (work in progress), October 2018.

Authors' Addresses

Darren Dukes (editor)
Cisco Systems
Canada

Email: ddukes@cisco.com

Clarence Filsfils
Cisco Systems
Belgium

Email: cfilsfil@cisco.com

Gaurav Dawra
LinkedIn
USA

Email: gdawra@linkedin.com

Xiaohu Xu
Alibaba
China

Email: xiaohu.xxh@alibaba-inc.com

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Pablo Camarillo Garvia
Cisco Systems
Spain

Email: pcamaril@cisco.com

Francois Clad
Cisco Systems
France

Stefano Salsano
Univ. of Rome Tor Vergata
Italy

Email: stefano.salsano@uniroma2.it

Network Working Group
Internet Draft
Intended status: Informational
Expires: January 2019

L. Dunbar
Huawei
Mehmet Toy
Verizon
July 2, 2018

Segment routing for SD-WAN paths over hybrid networks
draft-dunbar-sr-sdwan-over-hybrid-networks-02

Abstract

This document describes a method for end-to-end (E2E) SD-WAN paths (most likely encrypted) to traverse specific list of network segments, some of which are SR enabled and others may be IP networks that do not support SR, to achieve the desired optimal E2E quality.

The method described in this draft uses the principle of segment routing to enforce a SD-WAN path' head-end selected route traversing through a list of specific nodes of multiple network segments without requiring the nodes in each network segment to have the intelligence (or maintaining states) of selecting next hop or next domain. Those networks over which the SD-WAN path traverse have at least one SR enabled network, and some network segments (especially the last mile access portion) being existing IP networks (such as existing IPv4, IPv6 or others).

other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on January 2, 2009.

Table of Contents

1. Introduction

This document describes a method to enforce a SD-WAN path’s head-end selected route traversing through a list of specific nodes of multiple network segments without requiring the nodes in each network segments to have the intelligence (or maintaining states) of selecting next hop or next segments. Those networks over which the SD-WAN path traverse have at least one SR enabled network, and some network segments (especially the last mile access portion) being existing IP networks (such as existing IPv4, IPv6 or others).

SD-WAN, as described by ONUG (Open Network User Group), is about pooling WAN bandwidth from n service providers to get better WAN bandwidth management, visibility & control.

Throughout this document, the term "Classic SD-WAN" refers to a pair of CPEs in two locations aggregating N Service Providers’ paths, such as MPLS Paths and public internet paths. [SR-SD-WAN] describes using explicit routes within the SRv6 or SR-MPLS enabled networks to reach the desired quality for SD-WAN paths over the SRv6 or SR-MPLS enabled networks respectively.

Another way of using SD-WAN is for network service providers to extend its existing VPN to reach sites to which they do not have presence yet, with detailed use cases described in Section 3 of this document. Under this scenario, the SD-WAN path is laid over multiple hybrid network segments. This document focuses on this type of SD-WAN where a portion of SD-WAN path is over SR enabled networks and the other portion of the SD-WAN path is over existing IP networks, such as existing IPv4, LTE, etc. Under this scenario, the endpoints of the SD-WAN path (e.g. the CPE devices, one or both) are not directly attached to PEs of a SR domain.

The goal is to place a large portion of the SD-WAN path over a provider VPN to reach desired transport quality or making the SD-WAN path traversing specific ingress/egress PEs for the desired cost, quality or other reasons.

2. Definition of terms

Cloud DC: Off-Premises Data Centers that usually host applications and workload owned by different organizations or tenants.

Controller: Used interchangeably with SD-WAN controller to manage SD-WAN overlay path creation/deletion and monitoring the path conditions between two or more sites.

DMVPN: Dynamic Multipoint Virtual Private Network. DMVPN is a secure network that exchanges data between sites without needing to pass traffic through an organization’s headquarter virtual private network (VPN) server or router.

Heterogeneous Cloud: applications & workloads split among Cloud DCs owned & managed by different Cloud Providers.

Hybrid Cloud: applications & workloads split between on-premises Data centers and Cloud DCs. In this document Hybrid Cloud also include heterogeneous cloud as well.

SD-WAN: Software Defined Wide Area Network, which can mean many different things. In this document, "SD-WAN" refers to the solutions specified by ONUG (Open Network User Group), https://www.onug.net/software-defined-wide-area-network-sd-wan/, which is about pooling WAN bandwidth from n service providers to get better WAN bandwidth management, visibility & control.

SP: Network Service Provider

SR: Segment Routing

SR Domain: A domain that supports Segment Routing

VPC: Virtual Private Cloud. A service offered by many Cloud DC operators to allocate a logically isolated cloud resources, including compute, networking and storage.

3. Key Use Cases

3.1. SD-WAN Path over LTE network and SR Domain

MEF Cloud Service Architecture [MEF-Cloud] describes a use case of network operators needing to use SD-WAN over LTE for the last mile access that they do not have physical infrastructure, as shown below:

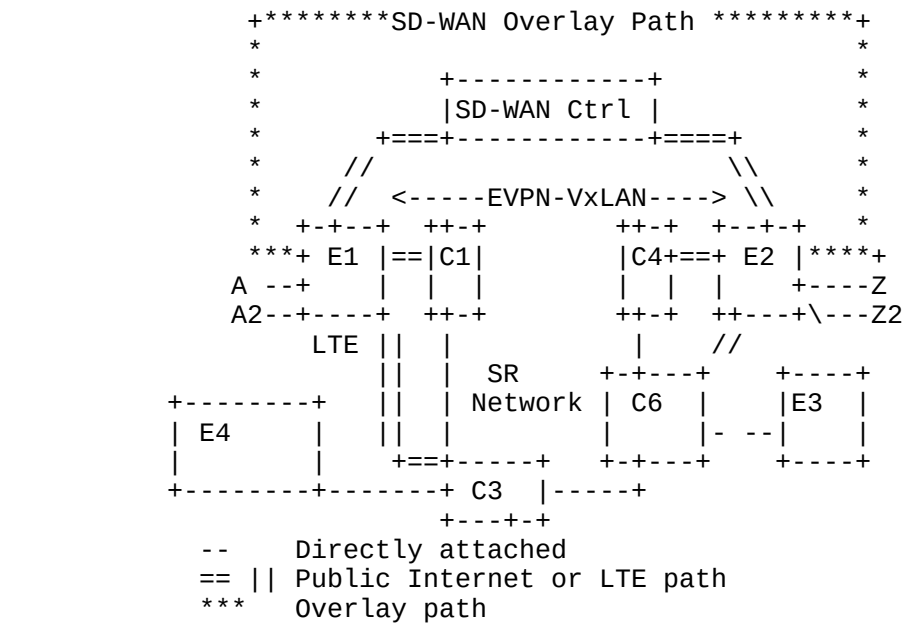


Figure 1: SD-WAN end points are attached to VPN via LTE

3.2. SD-WAN As Last Mile for Cloud DCs Access

Digital Transformation is propelling more and more enterprises to move their workloads/Apps to cloud DCs that are geographically close to their end users to improve end-to-end latency & overall user experience, or to comply with local data protection regulations. Conversely, workloads/Apps in those Cloud DCs can be easily shutdown when their end users' geographic base changes.

Because of the ephemeral property of the selected Cloud DCs, an enterprise or its network service provider may not have the direct links to the Cloud DCs that are optimal for hosting the enterprise's specific workloads/Apps. Under those circumstances, SD-WAN is a very flexible choice to interconnect the enterprise on-premises data centers & branch offices to its desired Cloud DCs.

However, SD-WAN paths over public internet can have unpredictable performance, especially over long distances and cross state/country boundaries. Therefore, it is highly desirable to place as much as possible the portion of SD-WAN paths over service provider VPN (e.g. enterprise's existing VPN) that have guaranteed SLA to minimize the distance/segments over public internet.

Under this scenario, one or both of the SD-WAN end points may not directly attached to the PEs of a SR Domain.

3.3. How & Why SR is useful for those use cases

Let us assume that the SD-WAN Controller is capable of computing optimal paths between two end-points (e.g. E1<->E2 in the Figure 2), either by communicating with the SR Domain controller/management-system, or by other methods which is out of the scope of this document.

The SR domain must have a set of PEs that have at least one port facing the external networks (such as the public internet or LTE termination).

Under this circumstance, SD-WAN end-points usually can reach multiple PEs.

In the diagram below, E1 <-> E2 SD-WAN (most likely IPsec encrypted tunnel) path can traverse C1 <-> C4, C1<->C6, C3<->C6, or C3<->C4 within the VPN. There are many flows (by different Apps) between E1 <-> E2. Some flows may need to traverse C1<->C4, others may need to traverse C3<->C6 or other segments within the VPN, which are determined by the SD-WAN controller based on the characteristics & need of the Apps, such as cost, available bandwidth, latency, or special functions only available at specific locations, etc.

Even with the same ingress/egress, some flows may need different segments across the SR Domain. It is not practical, or even possible, for PEs (e.g. C1, C2, C3 in this example) to determine which Apps' flows should egress C4 or C6 where both C4&C6 can reach E2.

Segment Routing can easily force the path to traverse the explicit egress node (C4 or C6), or explicit segments through the SR Domain based on the SLA requested by the SD-WAN head-end nodes.

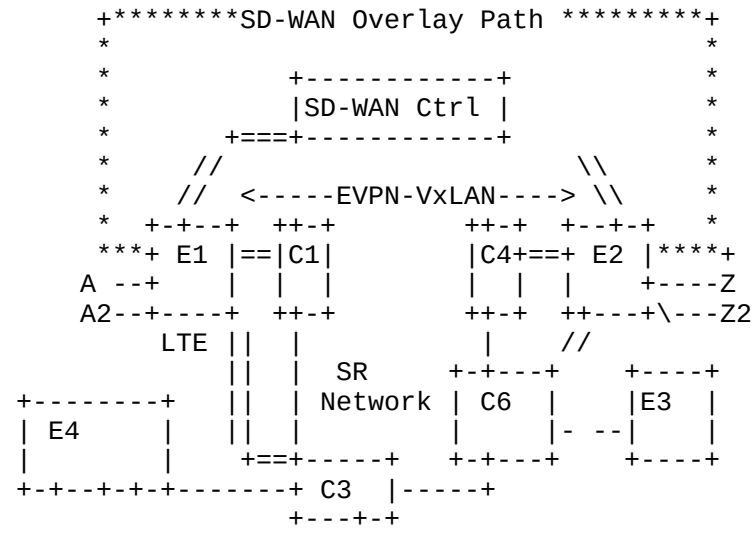


Figure 2: SDWAN end points not directly attached to PEs of SR Domain

4. Mechanism for SD-WAN path over one SR Domain and existing access

This section describes the mechanism to enforce a SD-WAN path' head-end selected route traversing through a list of specific nodes of multiple network segments without requiring the nodes in each network segment to have the intelligence (or maintaining states) of selecting next hop or next domain.

There may be two approaches here:
1) Controller installs the entire SID stack at E1.
2) Controller delivers to E1 a "Key" that the SR ingress PE can use to map to the SID stack for the packets arriving at the SR Ingress

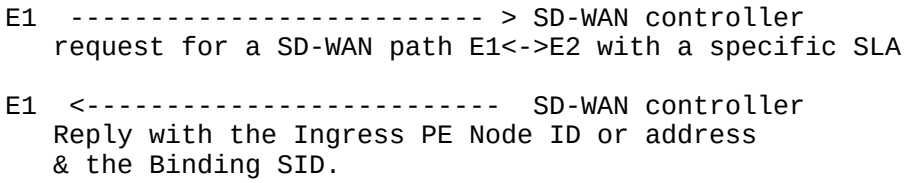
PE. Section 4.2 & 4.3 will describe how the "Key" is carried by the packets.

The Approach 1) requires less processing at the SR Ingress PE nodes, but only works if the remote CPEs are in the same Administrative domain as the SR domain. SR domain usually is not willing to expose its internal binding SIDs to devices in different administration domains. This approach also requires more changes to SD-WAN end nodes and need more header bytes added to the packets when traversing through 3rd party internet. Some SD-WAN nodes might not be capable of supporting encapsulating packets with the SID stack.

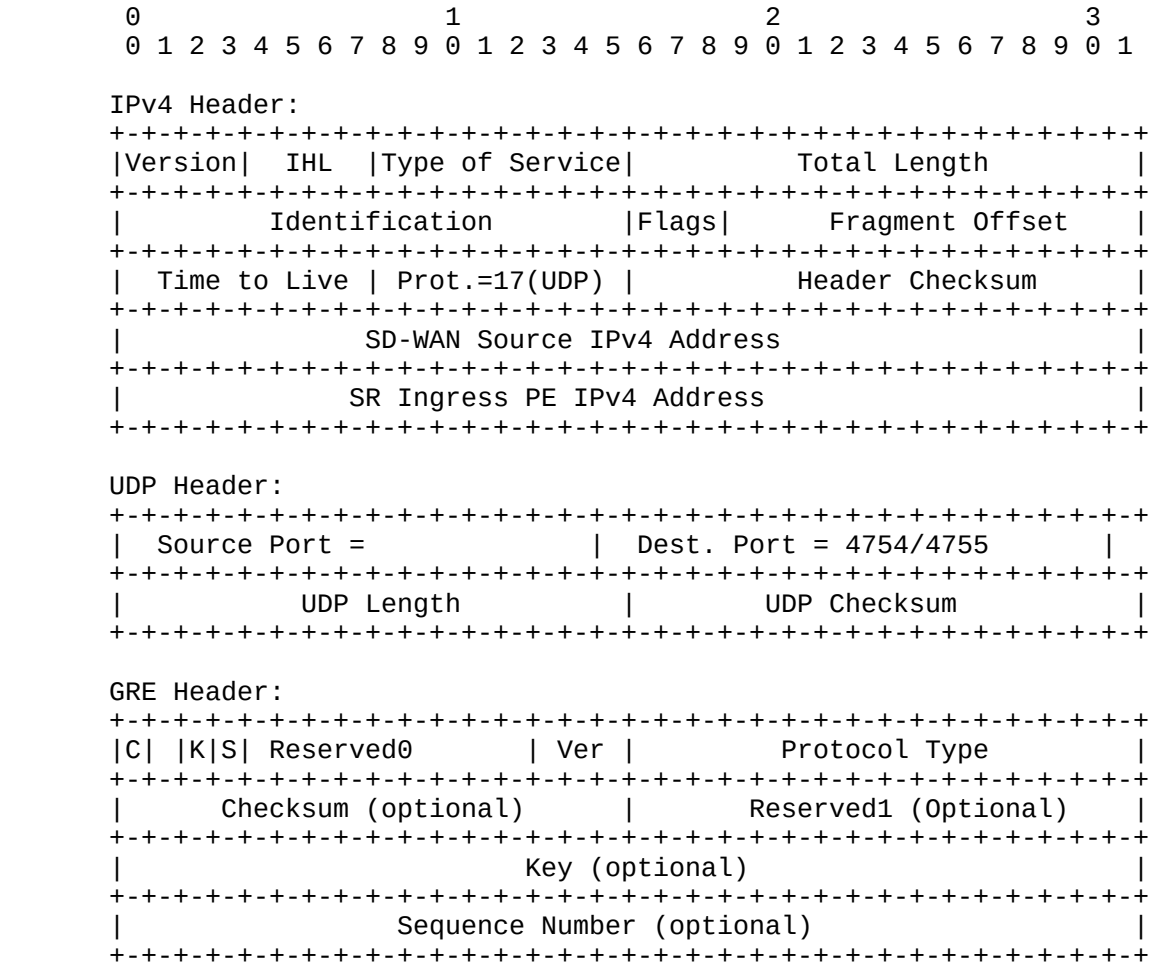
The Approach 2) above requires SR Ingress PE nodes to map the "Key" to the SID Stack and prepend the SID stack to the packets (Same processing for other traffic except the mapping is from the received "Key" carried in the payload).

4.1. Controller Delivers SID Stack to SD-WAN Head-end

This approach is straightforward.



Here is the packet header for SD-WAN Source Node to prepend to the payload:



To traverse SRv6 domain, SRv6 Header is appended after the GRE header [SRv6-SRH]:

To traverse MPLS-SR domain, a stack of MPLS labels is appended after GRE Header [MPLS-SR].

4.2. Using GRE Key to Differentiate Flows

This section describes a method of SD-WAN head-end node using GRE Key to indicate the desired property for different flows between SD-WAN end-points (E1<->E2 in the figure above): such as different desired routes through the SR Domain, different egress PEs based on cost, performance or other factors. It might be difficult or impossible to DiffServ bits carried by the packets to describe those flow properties.

The SR Domain ingress can map the GRE key to different SID through the SR Domain.

We assume that the SD-WAN Controller can determine which ingress PE can lead to the optimal path between E1<->E2. It is beyond the scope of this document on how SD-WAN controller computes the paths and how & what SD-WAN controller communicates with the SR Domain controller.

Here is the sequence of the flow:

```
E1  -------------------------- > SD-WAN controller
   request for a SD-WAN path E1<->E2 with a specific SLA

E1  <--------------------------  SD-WAN controller
   Reply with the Ingress PE Node ID or address
   & the GRE Key.
```

Note: the GRE key from the SD-WAN controller is for the ingress PE to correlate desired Path with the list of SIDs to prepend the packet across the SR domain.

When SD-WAN Controller get the E1<->E2 path request, it will communicate with the VPN Controller to get the optimal Ingress PE

Node ID (or IP address) and the GRE key to encapsulate the original packets between E1 <-> E2 (assuming IPsec Tunnel mode is used).

Upon receiving the GRE encapsulated packets, the provider ingress Edge C1/C3 decapsulates the outer GRE tunnel header, use the GRE key to map to the pre-defined (by the network controller) Binding SIDs, prepend the Binding SIDs to the packets, and forward its desired paths across the provider VPN.

Depending on how the SD-WAN path destination can be reached by the egress PE, the egress PE has different processing procedure:

- If the destination of the SD-WAN path is directly attached to the egress VPN PE node, the egress VPN PE decapsulates SR header and forward the packets to SD-WAN path destination node, such as the E2 in the figure above.
- If the destination of the SD-WAN path is IP reachable via IPv4 network from the egress VPN PE node, the egress VPN PE node decapsulates SR header and forward the packets to SD-WAN path destination node via its internet facing port to the SD-WAN path destination (i.e. the E2 node in the figure above).
- If the SD-WAN path is traversing multiple domains owned by different network operators, the egress PE processing is described in the next session.

4.3. Using UDP Source Port Number to Differentiate Flows

[RFC8086] describes how to use GRE-in-UDP source port number as entropy for better ECMP performance. When the remotely attached CPEs is within very close proximity to the PEs, e.g. only one or two hopes away like in LTE access, there is less issue if ECMP put all flows with same traffic classifier into one path. Then, those UDP numbers can also be used as a key to SR PE nodes to map to the appropriate SID to the packets.

Same as RFC8086, UDP source port values used as a key for SR PEs to map to appropriate SIDs SHOULD be chosen from the ephemeral port range (49152-65535) [RFC8085].

The GRE-in-UDP encapsulation format contains a UDP header [RFC768] and a GRE header [RFC2890]. The format is shown as follow (presented in bit order):

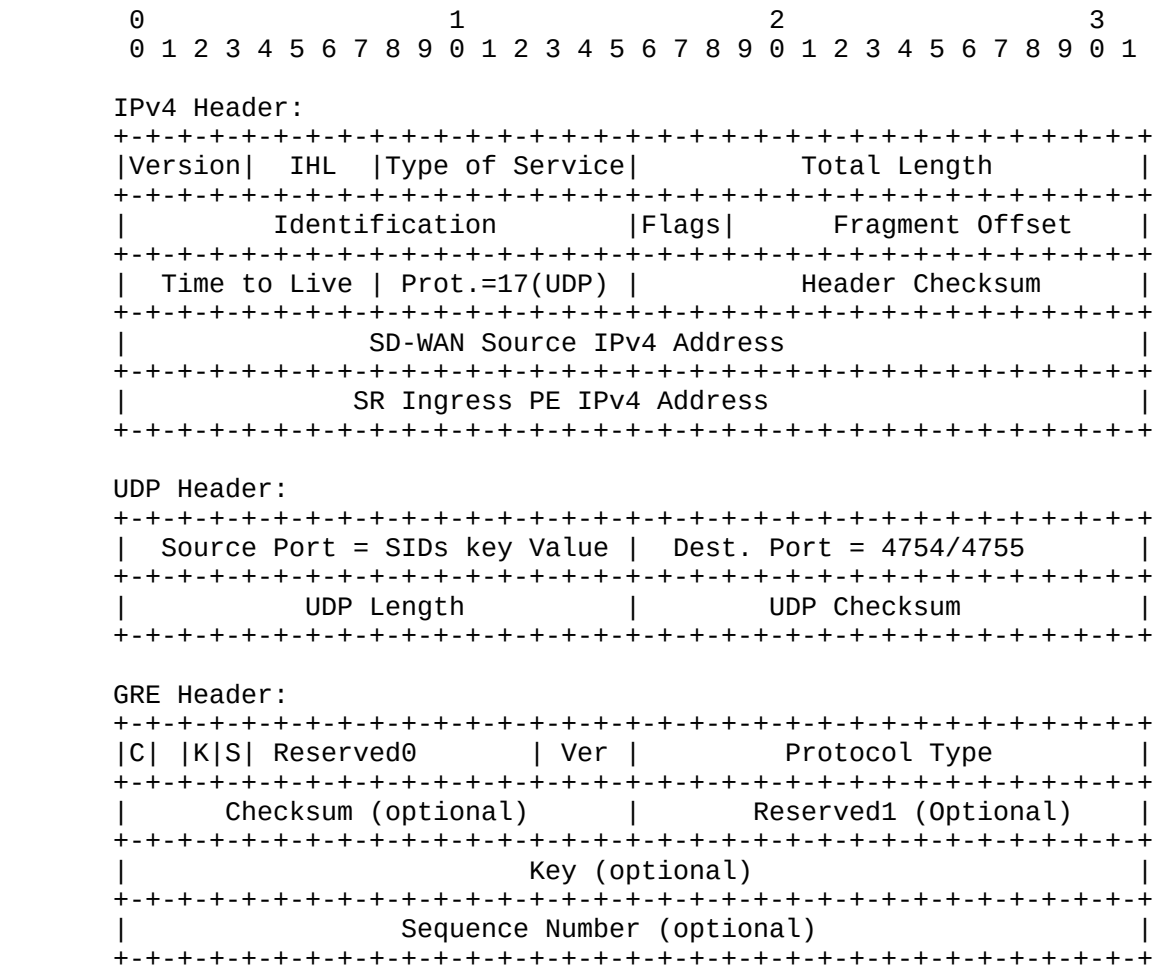


Figure 3: UDP + GRE Headers in IPv4

Here is the GRE Header for IPv6 network, i.e. the SD-WAN Source SD-WAN Destination, and SR PEs are all in IPv6 domain:

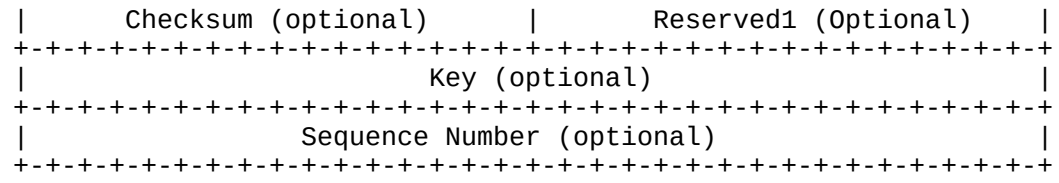
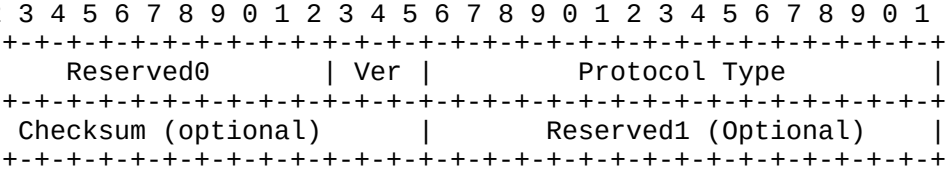


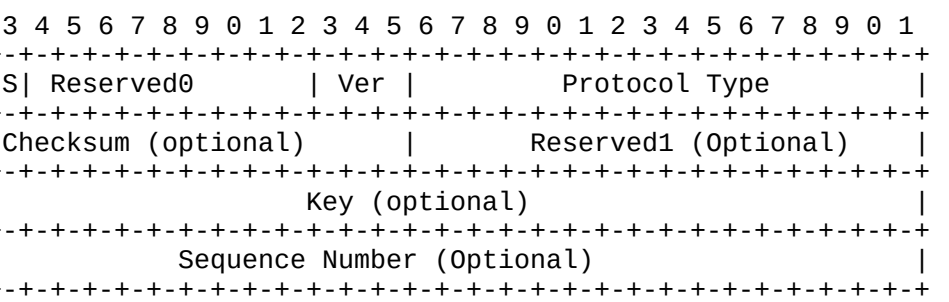
Figure 4: GRE+UDP for IPv6

4.4. GRE Header Extension

A new protocol type can be added to the GRE header [RFC2890] to make it easier for the SR PE to do the proper actions:



The proposed GRE header will have the following format:

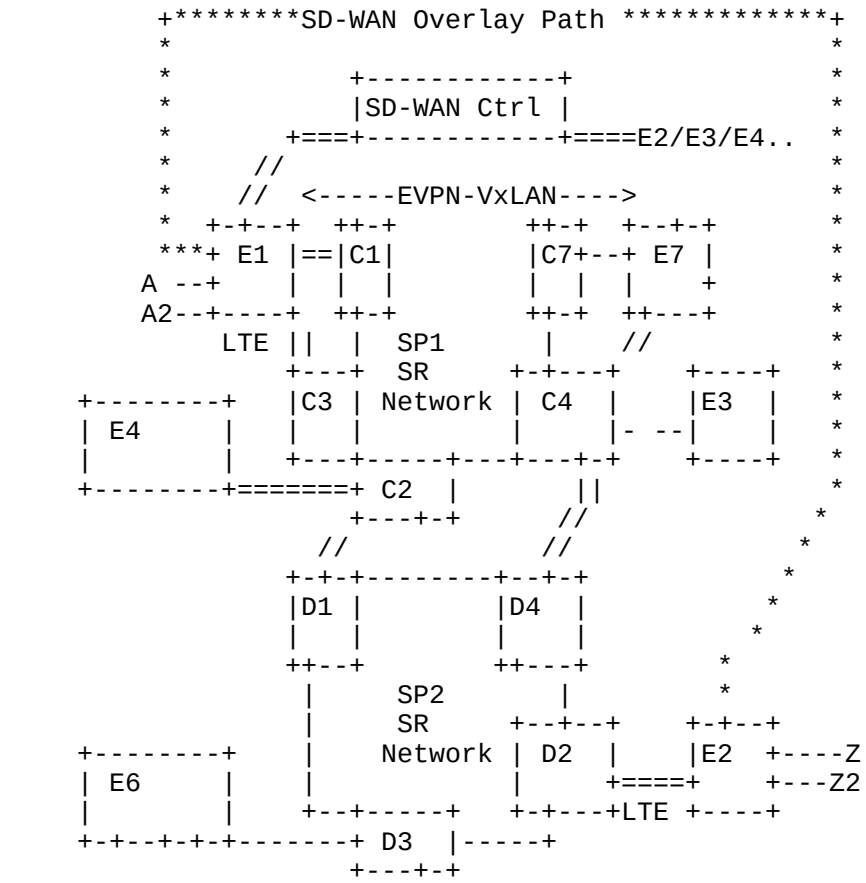


New protocol type (value to be assigned by IANA):
UDP-Key: Using UDP source port value as a Key for SR Ingress PE to map to the appropriate SIDs.

GRE-KEY: Using GRE Key value as a key for SR ingress PE to map to the appropriate SIDs

5. SD-WAN path over multiple SP managed domains

The following figure shows a SD-WAN Path E1<->E2 over two SP domains which are interconnected by public internet.



-- Directly attached
== || Public Internet or LTE path
** Overlay path

Figure 5: SD-WAN path over two different SP domains

Let’s assume that the SP-1 domain’s egress node for the SD-WAN path E1<->E2 is C2, which can reach D1 or D4 of SP-2 via public IP network (say IPv4 network).

Let’s also assume that the optimal route for some flows over SD-WAN path E1<->E2 are C1->C2->D1 and other flows are over C1->C2->D4 (out of the scope of this document on how the path is calculated).

If SP-1 is SR enabled, the mechanism described in Section 4 is applicable to the SD-WAN path source node E1 and the SP-1’s ingress PE (e.g. C1 or C3 in the figure).
However, the processing at egress node might be different depending on how the SP-1’s egress edges are connected to the SP-2’s ingress edge nodes.

5.1. When Both SP domains support SR

There may be three approaches here:

1) Controller installs the entire SID stack at E1, and the SID list contains SID entries belong to both domains.

2) Controller delivers to E1 the SID stack that only for the first domain, but delivers to C6 (egress node of first domain) the binding SID of the second domain.

3) Controller delivers a "Key" to E1, which can be encoded as GRE KEY or represented by the Source UDP port of the GRE encapsulation, for Ingress PE of the first SR Domain to map to its own SID stack as described in Section 4. The first SR Domain will reserve the "Key" through its domain and pass the "Key" to the second SR domain. The second SR Domain Ingress node will use the same method to map the "Key" to its SID stack.

5.2. When SP-2 does not support SR

Under this circumstance (which can be caused by SP-2 not supporting SR or not willing to share Binding SIDs to SP-1), if the packets arriving at SP-1 egress node C6 do not have any metadata indicating the types of encrypted payload, C6 does not really have much choice

other than simply forwarding the packets to E2 via public IP network. This way, the packets may or may not traverse through the SP-2 domain. If the distance between C6 and E2 is far, the quality of service can be unpredictable.

5.3. When SP-1 and SP-2 don't want to share network information

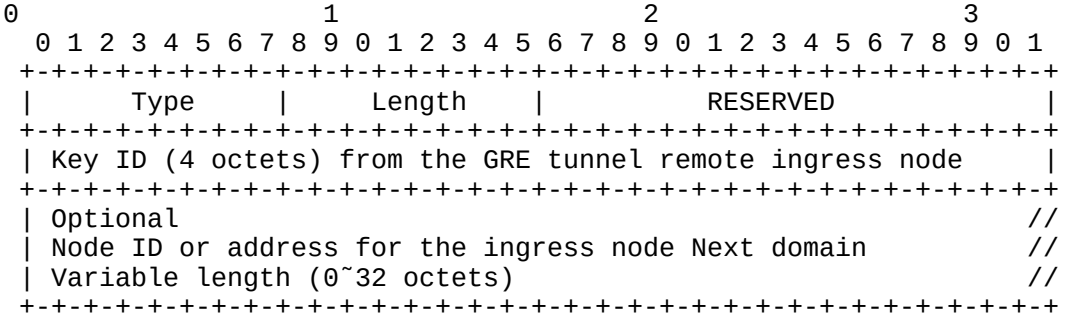
If SP-1's ingress node C1 can include the GRE KEY it receives from E1 in the data packets' SR header, the SP-1's egress node can map the Key to the SP-2's Ingress node and encapsulate the data packet in a new GRE header destined towards the SP-2's Ingress node. Then the SP-2's Ingress node can follow the procedure described in the Section 4 to forward the data packets across its domain.

If the first SR Domain does not support adding metadata to carry the "key" through its domain, the controller can deliver the "key" to SP-1's egress node the same time as it delivers the key to E1, knowing the SD-WAN path will need to traverse two domains with the second one does support SR but the two SPs don't want to exchange network information.

5.4. TLV to pass Metadata through SRv6 Domain

If SP-1 is SRv6 based, the ingress node C1 can append a TLV to the end of the SR Header [SRv6-SRH] to carry the KEY it receives from E1[Dc1].

The SP-1 egress node C6 can get the mapping between the KEYs and the Node-IDs (or Addresses) of the next domain's ingress edge node (i.e. D1 or D4 in the figure 3 above) from its network controller ahead of time.



TYPE: (to be assigned by IANA) is to indicate the TLV is for carrying the flow identifier of the packet encoded by the SD-WAN source node.

Upon receiving the packet, the egress node (C6) can

- find the Node-ID (or the address) for the next domain's ingress node,
- construct a GRE header with the Key received from the TLV above and the destination address from the mapping given by the controller,
- encapsulate the GRE header to the data packet (which has decapsulated SR header),
- and forward the packet to the public internet.

6. Security Considerations

Remotely attached CPEs might brought the following security risks:

1) Potential DDoS attack to the PEs with ports facing internet. I.e. the PE resource being attacked by unwanted traffic.
2) Potential risk of provider VPN network bandwidth being stolen by the entities who spoofed the addresses of SD-WAN end nodes.

To mitigate security risk of 1) above, it is absolutely necessary for PEs which accept remotely attached CPEs or simply have ports facing internet to enable Anti-DDoS feature to prevent major DDoS attack to those PEs.

To mitigate the security risk of 2) above, RFC7510 defines the use of DTLS to authenticate and encrypt the RFC7510 encapsulation.

However, for the scenario of SD-WAN source node being remotely attached to PEs, using the method recommended by RFC7510 means the source node has to perform DTLS on top of the IPSec encryption between SD-WAN end points E1<->E2. This can be too processing heavy for the SD-WAN end nodes. In addition, if there are many SD-WAN flows to traverse through the ingress PE (e.g. C1, C2, C4 in the figure 1 above), heavy processing is required on the ingress PEs.

Since the payload between E2<->E2 is already encrypted, the confidentiality of the payload is already ensured. The network operators need to balance between how much they can tolerant some percentage of bandwidth being stolen and how much extra cost they are willing to pay for completely prevent any unpaid traffic traversing through its VPN networks. For operators who opt for lower cost ingress PEs and CPEs, but can tolerant some percentage of bandwidth being used by unpaid subscribers, a simple approach can be considered:

- Embed a key in the packets, which can be changed periodically, like the digital signature used by a certificate authority or certification authority (CA).
- The key can be encoded in the GRE Key field between SD-WAN end node and Ingress PE. Since GRE has 24 bits, some fixed bits can be used to represent the signature of paid subscribers.

7. IANA Considerations

This document requires new protocol type:

Protocol type to be added to GRE header: SR_Route

8. References

8.1. Normative References

[RFC2890] G. Dommety "Key and Sequence Number Extensions to GRE". Sep. 2000.

8.2. Informative References

[RFC2735] B. Fox, et al "NHRP Support for Virtual Private networks". Dec. 1999.

[RFC8192] S. Hares, et al "Interface to Network Security Functions (I2NSF) Problem Statement and Use Cases", July 2017

[ITU-T-X1036] ITU-T Recommendation X.1036, "Framework for creation, storage, distribution and enforcement of policies for network security", Nov 2007.

[RFC6071] S. Frankel and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", Feb 2011.

[RFC4364] E. Rosen and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", Feb 2006

[RFC4664] L. Andersson and E. Rosen, "Framework for Layer 2 Virtual Private Networks (L2VPNs)", Sept 2006.

[SR-SD-WAN] D. Dukes, et al, "SR for SDWAN: VPN with Underlay SLA", draft-dukes-sr-for-sdwan-00, in progress, Oct 2017

[SRv6-SRH] S. Previdi, et al, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-13, in progress, April 2018.

[MPLS-SR] A. Bashandy, et al, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-13, in progress, April 2018.

[RFC7510] X. Xu, et al, "Encapsulating MPLS in UDP", April 2015.

[RFC8086] L. Yong, et al, "GRE-in-UDP Encapsulation", March 2017.

[MEF-Cloud] "Cloud Services Architecture Technical Specification", Work in progress, April 2018

9. Acknowledgments

Many thanks to Dean Cheng and Jim Guichard for the discussion and contributions.

Authors' Addresses

Linda Dunbar
Huawei
Email: Linda.Dunbar@huawei.com

Mehmet Toy
Verizon
One Verizon Way
Basking Ridge, NJ 07920
Email: mehmet.toy@verizon.com

SPRING Working Group
Internet-Draft
Intended status: Informational
Expires: April 24, 2019

C. Filsfils
K. Talaulikar, Ed.
Cisco Systems, Inc.
P. Krol
Google, Inc.
M. Horneffer
Deutsche Telekom
P. Mattes
Microsoft

October 21, 2018

SR Policy Implementation and Deployment Considerations
draft-filsfils-spring-sr-policy-considerations-02.txt

Abstract

Segment Routing (SR) allows a headend node to steer a packet flow along any path. Intermediate per-flow states are eliminated thanks to source routing. SR Policy framework enables the instantiation and the management of necessary state on the headend node for flows along a source routed paths using an ordered list of segments associated with their specific SR Policies. This document describes some of the implementation and deployment aspects that are useful for operationalizing the SR Policy architecture.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 24, 2019.

Table of Contents

1. Introduction

Segment Routing (SR) allows a headend node to steer a packet flow along any path. Intermediate per-flow states are eliminated with source routing [RFC8402].

The headend node steers a flow into a Segment Routing Policy (SR Policy) by augmenting packet headers with the ordered list of segments associated with that SR Policy. [I-D.ietf-spring-segment-routing-policy] defines the SR Policy architecture and details the concepts of SR Policy and steering into an SR Policy.

This document describes some of the implementation aspects for SR Policy framework which should be considered as suggestions. The same behavior, as defined in [I-D.ietf-spring-segment-routing-policy], may in fact be realized with other alternate approaches. The deployment aspects described in this document are also meant to only serve as guidelines. This document describes these aspects and other considerations related to SR Policy concepts as they are important to facilitate multi-vendor interoperable deployments for various SR Policy use-cases.

These apply equally to the MPLS [I-D.ietf-spring-segment-routing-mpls] and SRv6 [I-D.filsfils-spring-srv6-network-programming] instantiations of segment routing.

For reading simplicity, the illustrations are provided for the MPLS instantiations.

2. SR Policy Headend Architecture

This section provides a conceptual overview of components (or functions) that interact to implement SR Policy on a headend

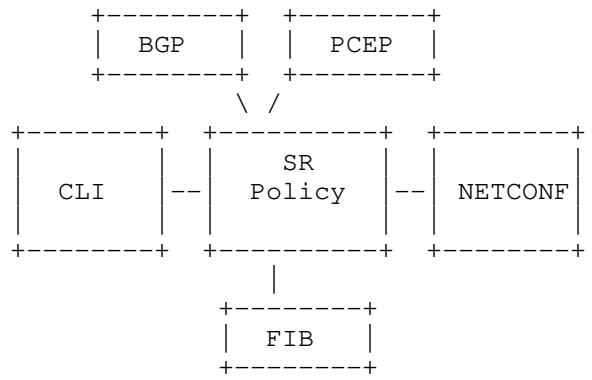


Figure 1: SR Policy Architecture at a Headend

The SR Policy functionality at a headend can be implemented in an SR Policy (SRP) process as illustrated in Figure 1 .

The SRP process interacts with other processes to learn candidate paths.

The SRP process selects the active path of an SR Policy.

The SRP process interacts with the RIB/FIB process to install an active SR Policy in the dataplane.

In order to validate explicit candidate paths and compute dynamic candidate paths, the SRP process maintains an SR Database (SR-DB) as specified in [I-D.ietf-spring-segment-routing-policy]. The SRP process interacts with other processes as shown in Figure 2 to collect the SR-DB information.

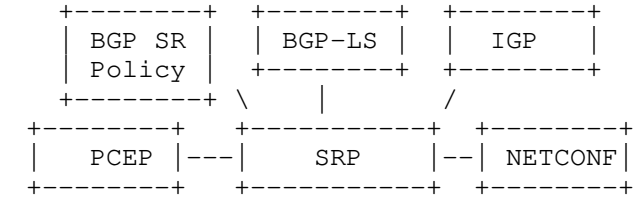


Figure 2: Topology/link-state database architecture

The SR Policy architecture supports both centralized and distributed control-plane.

3. Dynamic Path Computation

A dynamic candidate path for SR Policy is specified as an optimization objective and constraints and needs to be computed by either the headend or a Path Computation Element (PCE). The distributed or centralized computation aspect is described further in Section 5. This section describes the computation aspects of a dynamic path.

3.1. Optimization Objective

This document describes two optimization objectives:

- Min-Metric - requests computation of a solution Segment-List optimized for a selected metric.

- Min-Metric with margin and maximum number of SIDs - Min-Metric with two changes: a margin of by which two paths with similar metrics would be considered equal, a constraint on the max number of SIDs in the Segment-List.

The "Min-Metric" optimization objective requests to compute a solution Segment-List such that packets flowing through the solution Segment-List use ECMP-aware paths optimized for the selected metric. The "Min-Metric" objective can be instantiated for the IGP metric ([RFC1195] [RFC2328] [RFC5340]) xor the TE metric ([RFC5305] [RFC3630]) xor the latency extended TE metric ([RFC7810] [RFC7471]). This metric is called the O metric (the optimized metric) to distinguish it from the IGP metric. The solution Segment-List must be computed to minimize the number of SIDs and the number of Segment-Lists.

If the selected O metric is the IGP metric and the headend and tailend are in the same IGP domain, then the solution Segment-List is made of the single prefix-SID of the tailend.

When the selected O metric is not the IGP metric, then the solution Segment-List is made of prefix SIDs of intermediate nodes, Adjacency SIDs along intermediate links and potentially Binding SIDs (BSIDs) of intermediate policies.

In many deployments there are insignificant metric differences between mostly equal path (e.g. a difference of 100 usec of latency between two paths from NYC to SFO would not matter in most cases). The "Min-Metric with margin" objective supports such requirement.

The "Min-Metric with margin and maximum number of SIDs" optimization objective requests to compute a solution Segment-List such that packets flowing through the solution Segment-List do not use a path whose cumulative O metric is larger than the shortest-path O metric + margin.

If this is not possible because of the number of SIDs constraint, then one option is that the solution Segment-List minimizes the O metric while meeting the maximum number of SID constraints (i.e. path with the least value of O metric while using <= the number of SIDs specified). The other default option is to not come up with a solution unless the desired SLA is guaranteed.

Section 7 describes another approach for computing a solution Segment-List consisting of a single segment when the O metric is not the IGP metric by using the Flex Algorithm Prefix-SID of the tailend.

3.2. Constraints

The following constraints can be described:

- Inclusion and/or exclusion of TE affinity.

- Inclusion and/or exclusion of IP address.
- Inclusion and/or exclusion of SRLG.
- Inclusion and/or exclusion of admin-tag.
- Maximum accumulated metric (IGP, TE and latency).
- Maximum number of SIDs in the solution Segment-List.
- Maximum number of weighted Segment-Lists in the solution set.
- Diversity to another service instance (e.g., link, node, or SRLG disjoint paths originating from different head-ends).

3.3. SR Native Algorithm

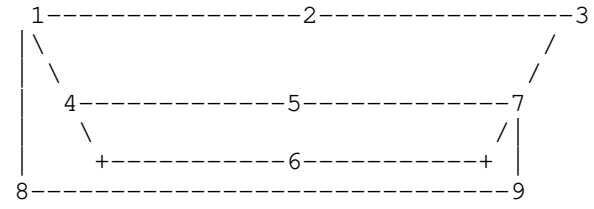


Figure 3: Illustration used to describe SR native algorithm

Let us assume that all the links have the same IGP metric of 10 and let us consider the dynamic path defined as: Min-Metric(from 1, to 3, IGP metric, margin 0) with constraint "avoid link 2-to-3".

A classical circuit implementation would do: prune the graph, compute the shortest-path, pick a single non-ECMP branch of the ECMP-aware shortest-path and encode it as a Segment-List. The solution Segment-List would be <4, 5, 7, 3>.

An SR-native algorithm would find a Segment-List that minimizes the number of SIDs and maximize the use of all the ECMP branches along the ECMP shortest path. In this illustration, the solution Segment-List would be <7, 3>.

In the vast majority of SR use-cases, SR-native algorithms should be preferred: they preserve the native ECMP of IP and they minimize the dataplane header overhead.

In some specific use-case (e.g. TDM migration over IP where the circuit notion prevails), one may prefer a classic circuit computation followed by an encoding into SIDs (potentially only using

non-protected Adj SIDs that pin the path to specific links and avoid ECMP to reflect the TDM paradigm).

SR-native algorithms are a local node behavior and are thus outside the scope of this document.

3.4. Path to SID

Let us assume the below diagram where all the links have an IGP metric of 10 and a TE metric of 10 except the link AB which has an IGP metric of 20 and the link AD which has a TE metric of 100. Let us consider the min-metric(from A, to D, TE metric, margin 0).

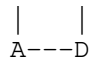


Figure 4: Illustration used to describe path to SID conversion

The solution path to this problem is ABCD.

This path can be expressed in SIDs as <B, D> where B and D are the IGP prefix SIDs respectively associated with nodes B and D in the diagram.

Indeed, from A, the IGP path to B is AB (IGP metric 20 better than ADCB of IGP metric 30). From B, the IGP path to D is BCD (IGP metric 20 better than BAD of IGP metric 30).

While the details of the algorithm remain a local node behavior, a high-level description follows: start at the headend and find an IGP prefix SID that leads as far down the desired path as possible(without using any link not included in the desired path). If no prefix SID exists, use the Adj SID to the first neighbor along the path. Restart from the node that was reached.

4. Candidate Path Selection

An SR Policy may have multiple candidate paths that are provisioned or signaled [I-D.ietf-idr-segment-routing-te-policy] [I-D.ietf-pce-segment-routing] from one of more sources. The tie-breaker rules defined in [I-D.ietf-spring-segment-routing-policy] result in determination of a single "active path" in a formal definition.

This section describe some examples for the candidate path selection based on the same rules.

Example 1:

Consider headend H where two candidate paths of the same SR Policy <color, endpoint> are signaled via BGP [I-D.ietf-idr-segment-routing-te-policy] and whose respective NLRIs have the same route distinguishers:

NLRI A with distinguisher = RD1, color = C, endpoint = N, preference P1.

NLRI B with distinguisher = RD1, color = C, endpoint = N, preference P2.

- Because the NLRIs are identical (same distinguisher), BGP will perform bestpath selection. Note that there are no changes to BGP best path selection algorithm.
- H installs one advertisement as bestpath into the BGP table.
- A single advertisement is passed to the SR Policy instantiation process.
- The SRP process does not perform any path selection.

Note that the candidate path's preference value does not have any effect on the BGP bestpath selection process.

Example 2:

Consider headend H where two candidate paths of the same SR Policy <color, endpoint> are signaled via BGP and whose respective NLRIs have different route distinguishers:

NLRI A with distinguisher = RD1, color = C, endpoint = N, preference P1.

NLRI B with distinguisher = RD2, color = C, endpoint = N, preference P2.

- Because the NLRIs are different (different distinguisher), BGP will not perform bestpath selection.
- H installs both advertisements into the BGP table.
- Both advertisements are passed to the SR Policy instantiation process.

- SRP process at H selects the candidate path advertised by NLRI B as the active path for the SR policy since P2 is greater than P1.

Note that the recommended approach is to use NLRIs with different distinguishers when several candidate paths for the same SR Policy (color, endpoint) are signaled via BGP to a headend.

Example 3:

Consider that a headend H learns two candidate paths of the same SR Policy <color, endpoint> one signaled via BGP and another via Local configuration.

NLRI A with distinguisher = RD1, color = C, endpoint = N, preference P1.

Local "foo" with color = C, endpoint = N, preference P2.

- H installs NLRI A into the BGP table.
- NLRI A and "foo" are both passed to the SRP process.
- SRP process at H selects the candidate path indicated by "foo" as the active path for the SR policy since P2 is greater than P1.

Now, let us consider cases, when an SR Policy has multiple valid candidate paths with the same best preference, the SRP process at a headend uses the rules described in [I-D.ietf-spring-segment-routing-policy] section 2.9 to select the active path. This is explained in the following examples:

Example 4:

Consider headend H with two candidate paths of the same SR Policy <color, endpoint> and the same preference value received from the same controller R and where RD2 is higher than RD1.

- NLRI A with distinguisher RD1, color C, endpoint N, preference P1(selected as active path at time t0).
- NLRI B with distinguisher RD2 (RD2 is greater than RD1), color C, endpoint N, preference P1 (passed to SR Policy instatiation process at time t1 > t0).

After t1, SRP process at H selects candidate path associated with NLRI B as active path of the SR policy since RD2 is higher than RD1. Here the time when the headend receives the candidate path via BGP is not a factor in the selection.

Note that, in such a scenario where there are redundant sessions to the same controller, the recommended approach is to use the same RD value for conveying the same candidate paths and let the BGP best path algorithm pick the best path.

Example 5:

Consider headend H with two candidate paths of the same SR Policy <color, endpoint> and the same preference value both received from the same controller R and where RD2 is higher than RD1.

Consider also that headend H is configured to override the discriminator tiebreaker specified in [I-D.ietf-spring-segment-routing-policy] section 2.9

- NLRI A with distinguisher RD1, color C, endpoint N, preference P1 (selected as active path at time t0).
- NLRI B with distinguisher RD2, color C, endpoint N, preference P1 (passed to SR Policy instatiation process at time t1).

Even after t1, SRP process at H retains candidate path associated with NLRI A as active path of the SR policy since the discriminator tiebreaker is disabled at H.

Example 6:

Consider headend H with two candidate paths of the same SR Policy <color, endpoint> and the same preference value.

- Local "foo" with color C, endpoint N, preference P1 (selected as active path at time t0).
- NLRI A with distinguisher RD1, color C, endpoint N, preference P1 (passed to SRP process at time t1).

Even after t1, SRP process at H retains candidate path associated with local candidate path "foo" as active path of the SR policy since

the Local protocol is preferred over BGP by default based on its higher protocol identifier value.

Example 7:

Consider headend H with two candidate paths of the same SR Policy <color, endpoint> and the same preference value but received via NETCONF from two controllers R and S (where S > R)

- Path A from R with distinguisher D1, color C, endpoint N, preference P1 (selected as active path at time t0).
- Path B from S with distinguisher D2, color C, endpoint N, preference P1 (passed to SRP process at time t1).

Note that the NETCONF process sends both paths to the SRP process since it does not have any tiebreaker logic. After t1, SRP process at H selects candidate path associated with Path B as active path of the SR policy.

5. Distributed and/or Centralized Control Plane

5.1. Distributed Control Plane within a single Link-State IGP area

Consider a single-area IGP with per-link latency measurement and advertisement of the measured latency in the extended-TE IGP TLV.

A head-end H is configured with a single dynamic candidate path for SR policy P with a low-latency optimization objective and endpoint E.

Clearly the SRP process at H learns the topology (and extended TE latency information) from the IGP and computes the solution Segment-List providing the low-latency path to E.

No centralized controller is involved in such a deployment.

The SR-DB at H only uses the Link-State DataBase (LSDB) provided by the IGP.

5.2. Distributed Control Plane across several Link-State IGP areas

Consider a domain D composed of two link-state IGP single-area instances (I1 and I2) where each sub-domain benefits from per-link latency measurement and advertisement of the measured latency in the related IGP. The link-state information of each IGP is advertised via BGP-LS [RFC7752] towards a set of BGP-LS route reflectors (RR).

H is a headend in IGP I1 sub-domain and E is an endpoint in IGP I2 sub-domain.

Using a BGP-LS session to any BGP-LS RR, H's SRP process may learn the link-state information of the remote domain I2. H can thus compute the low-latency path from H to E as a solution Segment-List that spans the two domains I1 and I2.

The SR-DB at H collects the LSDB from both sub-domains (I1 and I2).

No centralized controller is required.

5.3. Centralized Control Plane

Considering the same domain D as in the previous section, let us now assume that H does not have a BGP-LS session to the BGP-LS RR's. Instead, let us assume a controller "C" has at least one BGP-LS session to the BGP-LS RR's.

The controller C learns the topology and extended latency information from both sub-domains via BGP-LS. It computes a low-latency path from H to E as a Segment-List <S1, S2, S3> and programs H with the related explicit candidate path.

The headend H does not compute the solution Segment-List (it cannot). The headend only validates the received explicit candidate path. Most probably, the controller encodes the SID's of the Segment-List with Type-1. In that case, The headend's validation simply consists in resolving the first SID on an outgoing interface and next-hop.

The SR-DB at H only includes the LSDB provided by the IGP I1.

The SR-DB of the controller collects the LSDB from both sub-domains(I1 and I2).

5.4. Distributed and Centralized Control Plane

Consider the same domain D as in the previous section.

H's SRP process is configured to associate color C1 with a low-latency optimization objective.

H's BGP process is configured to steer a Route R/r of extended-color community C1 and of next-hop N via an SR policy (N, C1).

Upon receiving a first BGP route of color C1 and of next-hop N, H recognizes the need for an SR Policy (N, C1) with a low-latency objective to N. As N is outside the SRTE DB of H, H requests a

controller to compute such Segment-List (e.g., PCEP [I-D.ietf-pce-segment-routing]).

This is an example of hybrid control-plane: the BGP distributed control plane signals the routes and their TE requirements. Upon receiving these BGP routes, a local headend either computes the solution Segment-List (entirely distributed when the endpoint is in the SR-DB of the headend) else delegates the computation to a controller (hybrid distributed/centralized control-plane).

The SR-DB at H only includes the LSDB provided by the IGP.

The SR-DB of the controller collects the LSDB from both sub-domains.

6. Binding SID Aspects

The Binding SID (BSID) is fundamental to Segment Routing. It provides scaling, network opacity and service independence.

This section describes implementation and operational aspects related to the Binding SID.

6.1. Benefits of Binding SID

A simplified illustration is provided on the basis of Figure 5 where it is assumed that S, A, B, Data Center Interconnect DCI1 and DCI2 share the same IGP-SR instance in the data-center 1 (DC1). DCI1, DCI2, C, D, E, F, G, DCI3 and DCI4 share the same IGP-SR domain in the core. DCI3, DCI4, H, K and Z share the same IGP-SR domain in the data-center 2 (DC2).

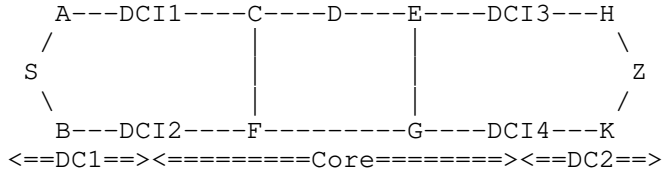


Figure 5: A Simple Datacenter Topology

In this example, it is assumed no redistribution between the IGP's and no presence of BGP-LU. The inter-domain communication is only provided by SR through SR Policies.

The latency from S to DCI1 equals to DCI2. The latency from Z to DCI3 equals to DCI4. All the intra-DC links have the same IGP metric 10.

The path DCI1, C, D, E, DCI3 has a lower latency and lower capacity than the path DCI2, F, G, DCI4.

The IGP metrics of all the core links are set to 10 except the links D-E which is set to 100.

A low-latency multi-domain policy from S to Z may be expressed as <DCI1, BSID, Z> where:

- DCI1 is the prefix SID of DCI1.
- BSID is the Binding SID bound to an SR policy <D, D2E, DCI3> instantiated at DCI1.
- Z is the prefix SID of Z.

Without the use of an intermediate core SR Policy (efficiently summarized by a single BSID), S would need to steer its low-latency flow into the policy <DCI1, D, D2E, DCI3, Z>.

The use of a BSID (and the intermediate bound SR Policy) decreases the number of segments imposed by the source.

A BSID acts as a stable anchor point which isolates one domain from the churn of another domain. Upon topology changes within the core of the network, the low-latency path from DCI1 to DCI3 may change. While the path of an intermediate policy changes, its BSID does not change. Hence the policy used by the source does not change, hence the source is shielded from the churn in another domain.

A BSID provides opacity and independence between domains. The administrative authority of the core domain may not want to share information about its topology. The use of a BSID allows keeping the service opaque. S is not aware of the details of how the low-latency service is provided by the core domain. S is not aware of the need of the core authority to temporarily change the intermediate path.

6.2. Centralized Discovery of available BSID

This section explains how controllers can discover the local SIDs available at a node N so as to pick an explicit BSID for a SR Policy to be instantiated at headend N.

Any controller can discover the following properties of a node N (e.g., via BGP-LS , NETCONF etc.):

- its local topology [RFC7752].
- its topology-related SIDs (Prefix SIDs, Adj SID and EPE SID [I-D.ietf-idr-bgp-ls-segment-routing-ext] [I-D.ietf-idr-bgpls-segment-routing-epe]).
- its Segment Routing Label Block (SRLB).
- its SR Policies and their BSID ([I-D.ietf-pce-segment-routing] [I-D.sivabalan-pce-binding-label-sid] [I-D.ietf-idr-te-lsp-distribution]).

Any controller can thus infer the available SIDs in the SRLB of any node with the assumption that all SIDs allocated from the SRLB on that node are being advertised by it via some protocols or mechanisms to the controller.

As an example, a controller discovers the following characteristics of N: SRLB (4000, 8000), 3 Adj SIDs (4001, 4002, 4003), 2 EPE SIDs (4004, 4005) and 3 SRTE policies (whose BSIDs are respectively 4006, 4007 and 4008). This controller can deduce that the SRLB sub-range (4009, 8000) is free for allocation.

A controller is not restricted to use the next numerically available SID in the available SRLB sub-range. It can pick any label in the subset of available labels. This random pick make the chance for a collision unlikely.

An operator could also sub-allocate the SRLB between different controllers (e.g. (4000-4499) to controller 1 and (4500-5000) to controller 2).

Inter-controller state-synchronization may be used to avoid/detect collision in BSID.

All these techniques make the likelihood of a collision between different controllers very unlikely.

In the unlikely case of a collision, the controllers will detect it through system alerts, BGP-LS reporting using [I-D.ietf-idr-te-lsp-distribution] or PCEP notification [RFC8231]. They then have the choice to continue the operation of their SR Policy with the dynamically allocated BSID or re-try with another explicit pick.

Note: in deployments where PCE Protocol (PCEP) is used between head-end and controller (PCE), a head-end can report BSID as well as policy attributes (e.g., type of disjointness) and operational and administrative states to controller. Similarly, a controller can also assign/update the BSID of a policy via PCEP when instantiating or updating SR Policy.

7. Flex-Algorithm Based SR Policies

SR allows for association of algorithms to Prefix SIDs [RFC8402]. [I-D.ietf-lsr-flex-algo] defines the IGP based Flex-Algorithm solution which allows IGPs themselves to compute constraint based paths over the network. Prefix SIDs for the specific flex-algorithm and associated with a node are used in the forwarding plane to steer along the specific constraint path to that node.

As specified in [RFC8402] these IGP Flex Algo Prefix SIDs can be used as segments within SR Policies thereby leveraging the underlying IGP Flex Algo solution.

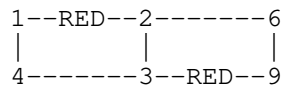


Figure 6: Illustration for Flex-Alg SID

Now let us assume that

- 1, 2, 3 and 4 are part of IGP 1.
- 2, 6, 9 and 3 are part of IGP 2.
- All the IGP link costs are 10.
- Links 1to2 and 3to9 are colored with IGP Link Affinity Red.
- Flex-Alg1 is defined in both IGPs as: avoid red, minimize IGP metric.
- All nodes of each IGP domain are enabled for FlexAlg1
- SID(k, 0) represents the PrefixSID of node k according to Alg=0.
- SID(k, FlexAlg1) represents the PrefixSID of node k according to Flex-Alg1.

A controller can steer a flow from 1 to 9 through an end-to-end path that avoids the RED links of both IGP domains thanks to the explicit SR Policy <SID(2, FlexAlg1), SID9(FlexAlg1)>.

8. Layer 2 and Optical Transport

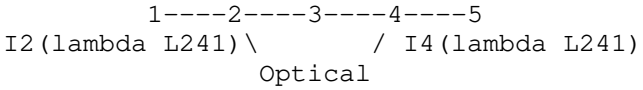


Figure 7: SR Policy with integrated DWDM

An explicit candidate path can express a path through a transport layer beneath IP (ATM, FR, DWDM). The transport layer could be ATM, FR, DWDM, back-to-back Ethernet etc. The transport path is modelled as a link between two IP nodes with the specific assumption that no distributed IP routing protocol runs over the link. The link may have IP address or be IP unnumbered. Depending on the transport protocol case, the link can be a physical DWDM interface and a lambda (integrated solution), an Ethernet interface and a VLAN, an ATM interface with a VPI/VCI, a FR interface with a DLCI etc.

Using the DWDM integrated use-case of Figure 7 as an illustration, let us assume

- nodes 1, 2, 3, 4 and 5 are IP routers running an SR-enable IGP on the links 1-2, 2-3, 3-4 and 4-5.
- The SRGB is homogeneous (16000, 24000).
- Node K's prefix SID is 16000+K.
- node 2 has an integrated DWDM interface I2 with Lambda L1.
- node 4 has an integrated DWDM interface I4 with Lambda L2.
- the optical network is provisioned with a circuit from 2 to 4 with continuous lambda L241 (details outside the scope of this document).
- Node 2 is provisioned with an SR policy with Segment-List <I2(L241)> and Binding SID B where I2(L241) is of type 5 (IPv4) or type 7 (IPv6), see section 4 of [I-D.ietf-spring-segment-routing-policy] .
- node 1 steers a packet P1 towards the prefix SID of node 5 (16005).

o node 1 steers a packet P2 on the SR policy <16002, B, 16005>.

In such a case, the journey of P1 will be 1-2-3-4-5 while the journey of P2 will be 1-2-lambda(L241)-4-5. P2 skips the IP hop 3 and leverages the DWDM circuit from node 2 to node 4. P1 follows the shortest-path computed by the distributed routing protocol. The path of P1 is unaltered by the addition, modification or deletion of optical bypass circuits.

The salient point of this example is that the SR Policy architecture seamlessly support explicit candidate paths through any transport sub-layer.

BGP-LS Extensions to describe the sub-IP-layer characteristics of the SR Policy are out of scope of this document (e.g. in Figure 7, the DWDM characteristics of the SR Policy at node 2 in terms of latency, loss, security, domain/country traversed by the circuit etc.).

Further details of the SR Policy use-case for Packet Optical networks are specified in [I-D.anand-spring-poi-sr] .

9. Security Considerations

The security considerations related to Segment Routing architecture are described in [RFC8402] and for SR Policy architecture are described in [I-D.ietf-spring-segment-routing-policy] and they apply to this document as well.

10. IANA Considerations

This document has no actions for IANA.

11. Acknowledgement

The authors like to thank Tarek Saad, Dhanendra Jain, Muhammad Durrani and Rob Shakir for their valuable comments and suggestions.

12. Contributors

The following people have contributed to this document:

Siva Sivabalan
Cisco Systems
Email: msiva@cisco.com

Zafar Ali
Cisco Systems
Email: zali@cisco.com

Jose Liste
Cisco Systems
Email: jliste@cisco.com

Francois Clad
Cisco Systems
Email: fclad@cisco.com

Kamran Raza
Cisco Systems
Email: skraza@cisco.com

Shraddha Hegde
Juniper Networks
Email: shraddha@juniper.net

Steven Lin
Google, Inc.
Email: stevenlin@google.com

Alex Bogdanov
Google, Inc.
Email: bogdanov@google.com

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

Dirk Steinberg
Steinberg Consulting
Email: dws@steinbergnet.net

Bruno Decraene
Orange Business Services
Email: bruno.decraene@orange.com

Stephane Litkowski
Orange Business Services
Email: stephane.litkowski@orange.com

Luay Jalil
Verizon
Email: luay.jalil@verizon.com

13. References

13.1. Normative References

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-01 (work in progress), June 2018.

[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>.

13.2. Informative References

[I-D.anand-spring-poi-sr]
Anand, M., Bardhan, S., Subrahmaniam, R., Tantsura, J., Mukhopadhyaya, U., and C. Filsfils, "Packet-Optical Integration in Segment Routing", draft-anand-spring-poi-sr-06 (work in progress), July 2018.

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-05 (work in progress), July 2018.

[I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H., and M. Chen, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-10 (work in progress), October 2018.

[I-D.ietf-idr-bgpls-segment-routing-epe]
Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", draft-ietf-idr-bgpls-segment-routing-epe-17 (work in progress), October 2018.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Jain, D., Mattes, P., Rosen, E., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-04 (work in progress), July 2018.

[I-D.ietf-idr-te-lsp-distribution]
Previdi, S., Talaulikar, K., Dong, J., Chen, M., Gredler, H., and J. Tantsura, "Distribution of Traffic Engineering (TE) Policies and State using BGP-LS", draft-ietf-idr-te-lsp-distribution-09 (work in progress), June 2018.

[I-D.ietf-lsr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-algo-00 (work in progress), May 2018.

[I-D.ietf-pce-segment-routing]
Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "PCEP Extensions for Segment Routing", draft-ietf-pce-segment-routing-14 (work in progress), October 2018.

[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-14 (work in progress), June 2018.

[I-D.sivabalan-pce-binding-label-sid]
Sivabalan, S., Filsfils, C., Tantsura, J., Hardwick, J., Previdi, S., and D. Dhody, "Carrying Binding Label/ Segment-ID in PCE-based Networks.", draft-sivabalan-pce-binding-label-sid-05 (work in progress), October 2018.

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <https://www.rfc-editor.org/info/rfc1195>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <https://www.rfc-editor.org/info/rfc3630>.

[RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <https://www.rfc-editor.org/info/rfc5305>.

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

[RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <https://www.rfc-editor.org/info/rfc7471>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC7810] Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 7810, DOI 10.17487/RFC7810, May 2016, <https://www.rfc-editor.org/info/rfc7810>.

[RFC8231] Crabbe, E., Minei, I., Medved, J., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE", RFC 8231, DOI 10.17487/RFC8231, September 2017, <https://www.rfc-editor.org/info/rfc8231>.

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Pegasus Parc
De kleetlaan 6a, DIEGEM BRABANT 1831
BELGIUM

Email: cfilsfil@cisco.com

Ketan Talaulikar (editor)
Cisco Systems, Inc.

Email: ketant@cisco.com

Przemyslaw Krol
Google, Inc.

Email: pkrol@google.com

Martin Horneffer
Deutsche Telekom

Email: martin.horneffer@telekom.de

Paul Mattes
Microsoft
One Microsoft Way
Redmond, WA 98052-6399
USA

Email: pamattes@microsoft.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 9, 2018

C. Filsfils
Z. Ali, Ed.
Cisco Systems, Inc.
M. Horneffer
Deutsche Telekom
D. Voyer
Bell Canada
M. Durrani
Equinix
R. Raszuk
Bloomberg LP

June 7, 2018

Segment Routing Traffic Accounting Counters
draft-filsfils-spring-sr-traffic-counters-00.txt

Abstract

Segment Routing (SR) allows a headend node to steer a packet flow along any path. Intermediate per-flow states are eliminated thanks to source routing. Traffic accounting plays a critical role in network operation. A traffic account solution is required for SR networks that provides the necessary functionality without creating any additional per SR path states in the fabric.

This document describes counters for traffic accounting in SR networks.

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 December 9, 2018.

Table of Contents

1. Introduction

This document defines counters for traffic accounting in segment routing (SR) [I-D.ietf-spring-segment-routing] networks. The essence of Segment Routing consists in scaling the network by only maintaining per-flow state at the source or edge of the network. Specifically, only the headend of an SR policy [I-D.filsfils-spring-segment-routing-policy] maintains the related per-policy state. Egress and midpoints along the source route do not maintain any per-policy state. The traffic counters described in this section respects the architecture principles of SR, while given visibility to the service provider for network operation and capacity planning.

This document specifies prefix-SID, interface and SR Policy counters to be implemented at each SR router. Furthermore, it describes the traffic matrix (TM) counters for accounting at the TM border routers (details described in Section 2.4).The goal of the document is to specify these necessary counters for traffic accounting in an SR network. The actual usage of this information and leveraging for various use-cases is outside the scope of this document. [I-D.ali-spring-sr-traffic-accounting] describes some of the use-cases and application of these counters in an SR network.

This document assumes that the routers export the traffic counters defined in Section 2 to an external controller. The methods for collection of this information by the controller is beyond the scope of the document.

2. SR Traffic Counters

2.1. Traffic Counters Naming convention

The section uses the following naming convention when referring to the various counters. This is done in order to assign mnemonic names to SR counters.

- The term counter(s) in all of the definitions specified in this document refers either to the (packet, byte) counters or the byte counter.
- SR: any traffic whose FIB lookup is a segment (IGP prefix/Adj segments, BGP segments, any type of segments) or the matched FIB entry is steered on an SR Policy.
- INT in name indicates a counter is implemented at a per interface level.
- E in name refers to egress direction (with respect to the traffic flow).
- I in name refers to ingress direction (with respect to the traffic flow).
- TC in name indicates a counter is implemented on a Traffic Class (TC) basis.
- TM in name refers to a Traffic Matrix (TM) counter.
- PRO in name indicates that the counter is implemented on per protocol/adjacency type basis. Per PRO counters in this document can either be accounts for:
  - LAB (Labelled Traffic): the matched FIB entry is a segment, and the outgoing packet has at least one label (that label does not have to be a segment label, e.g., the label may be a VPN label).
  - V4 (IPv4 Traffic): the matched FIB entry is a segment which is PoP'ed. The outgoing packet is IPv4.

    * V6 (IPv6 Traffic): the matched FIB entry is a segment which is PoP'ed. The outgoing packet is IPv6.

- POL in name refers to a Policy counter.
- BSID in name indicates a policy counter for labelled traffic.
- SL in name indicates a policy counter is implemented at a Segment-List (SL) level.

Counter nomenclature is exemplified using the following example:

- SR.INT.E.PRO: Per-interface per-protocol aggregate egress SR traffic.
- POL.BSID: Per-SR Policy labelled steered aggregate traffic counter.

2.2. Per-Interface SR Counters

For each local interface, node N maintains the following per-interface SR counters. These counters include accounting due to push, pop or swap operations on SR traffic.

2.2.1. Per interface, per protocol aggregate egress SR traffic counters (SR.INT.E.PRO)

The following counters are included under this category.

- SR.INT.E.LAB: For each egress interface (INT.E), N MUST maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as labelled traffic.
- SR.INT.E.V4: For each egress interface (INT.E), N MUST maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as IPv4 traffic (due to the pop operation).
- SR.INT.E.V6: For each egress interface (INT.E), N MUST maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as IPv6 traffic (due to the pop operation).

2.2.2. Per interface, per traffic-class, per protocol aggregate egress SR traffic counters (SR.INT.E.PRO.TC)

This counter provides per Traffic Class (TC) breakdown of SR.INT.E.PRO. The following counters are included under this category.

- SR.INT.E.LAB.TC: For each egress interface (INT.E) and a given Traffic Class (TC), N SHOULD maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as labelled traffic.

- SR.INT.E.V4.TC: For each egress interface (INT.E) and a given Traffic Class (TC), N SHOULD maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as IPv4 traffic (due to the pop operation).

- SR.INT.E.V6.TC: For each egress interface (INT.E) and a given Traffic Class (TC), N SHOULD maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as IPv6 traffic (due to the pop operation).

2.2.3. Per interface aggregate ingress SR traffic counter (SR.INT.I)

The SR.INT.I counter is defined as follows:

For each ingress interface (INT.I), N SHOULD maintain counter(s) for the aggregate SR traffic received on I.

2.2.4. Per interface, per TC aggregate ingress SR traffic counter (SR.INT.I.TC)

This counter provides per Traffic Class (TC) breakdown of the SR.INT.I. It is defined as follow:

For each ingress interface (INT.I) and a given Traffic Class (TC), N MAY maintain counter(s) for the aggregate SR traffic (matching the traffic class TC criteria) received on I.

2.3. Prefix SID Counters

For a remote prefix SID S, node N maintains the following prefix SID counters. These counters include accounting due to push, pop or swap operations on the SR traffic.

2.3.1. Per-prefix SID egress traffic counter (PSID.E)

This counter is defined as follows:

For a remote prefix SID S, N MUST maintain counter(s) for aggregate traffic forwarded towards S.

2.3.2. Per-prefix SID per-TC egress traffic counter (PSID.E.TC)

This counter provides per Traffic Class (TC) breakdown of PSID.E. It is defined as follows:

For a given Traffic Class (TC) and a remote prefix SID S, N SHOULD maintain counter(s) for traffic forwarded towards S.

2.3.3. Per-prefix SID, per egress interface traffic counter (PSID.INT.E)

This counter is defined as follows:

For a given egress interface (INT.E) and a remote prefix SID S, N SHOULD maintain counter(s) for traffic forwarded towards S over the (INT.E) interface.

2.3.4. Per-prefix SID per TC per egress interface traffic counter (PSID.INT.E.TC)

This counter provides per Traffic Class (TC) breakdown of PSID.INT.E. It is defined as follows:

For a given Traffic Class (TC), an egress interface (INT.E) and a remote prefix SID S, N MAY maintain counter(s) for traffic forwarded towards S over the (INT.E) interface.

2.3.5. Per-prefix SID, per ingress interface traffic counter (PSID.INT.I)

This counter is defined as follows:

For a given ingress interface (INT.I) and a remote prefix SID S, N MAY maintain counter(s) for the traffic received on I and forwarded towards S.

2.3.6. Per-prefix SID, per TC, per ingress interface traffic counter (PSID.INT.I.TC)

This counter provides per Traffic Class (TC) breakdown of PSID.INT.I. It is defined as follows:

For a given Traffic Class (TC), ingress interface (INT.I), and a remote prefix SID S, N MAY maintain counter(s) for the traffic received on I and forwarded towards S.

2.4. Traffic Matrix Counters

A traffic matrix T(N, M) is the amount of traffic entering the network at node N and leaving the network at node M, where N and M are border nodes at an arbitrarily defined boundary in the network [Traffic-Matrices] is. The TM border defines the arbitrary boundary nodes of a contiguous portion of the network across which service providers wish to measure traffic flows. The traffic matrix (also called demand matrix) contains all the demands crossing the TM border. It has as many rows as ingress edge nodes and as many columns as egress edge nodes at the TM border. The demand D(N, M) is the cell of the matrix at row N and column M. In other words, a Traffic Matrix provides, for every ingress point N into the network and every egress point M out of the network, the volume of traffic T(N, M) from N to M over a given time interval. To measure the traffic matrix, nodes in an SR network designate its interfaces as either internal or external.

When Node N receives a packet destined to remote prefix SID M, N maintains the following counters. These counters include accounting due to push, pop or swap operations.

2.4.1. Per-Prefix SID Traffic Matrix counter (PSID.E.TM)

This counter is defined as follows:

For a given remote prefix SID M, N SHOULD maintain counter(s) for all the traffic received on any external interfaces and forwarded towards M.

2.4.2. Per-Prefix, Per TC SID Traffic Matrix counter (PSID.E.TM.TC)

This counter provides per Traffic Class (TC) breakdown of PSID.E.TM. It is defined as follows:

For a given Traffic Class (TC) and a remote prefix SID M, N SHOULD maintain counter(s) for all the traffic received on any external interfaces and forwarded towards M.

2.5. SR Policy Counters

Per policy counters are only maintained at the policy head-end node. For each SR policy [I-D.filsfils-spring-segment-routing-policy] , the head-end node maintains the following counters.

2.5.1. Per-SR Policy Aggregate traffic counter (POL)

This counter includes both labelled and unlabelled steered traffic. It is defined as:

For each SR policy (P), head-end node N MUST maintain counter(s) for the aggregate traffic steered onto P.

2.5.2. Per-SR Policy labelled steered aggregate traffic counter (POL.BSID)

This counter is defined as:

For each SR policy (P), head-end node N SHOULD maintain counter(s) for the aggregate labelled traffic steered onto P. Please note that labelled steered traffic refers to incoming packets with an active SID matching a local BSID of an SR policy at the head-end.

2.5.3. Per-SR Policy, per TC Aggregate traffic counter (POL.TC)

This counter provides per Traffic Class (TC) breakdown of POL. It is defined as follows:

For each SR policy (P) and a given Traffic Class (TC), head-end node N SHOULD maintain counter(s) for the aggregate traffic (matching the traffic class TC criteria) steered onto P.

2.5.4. Per-SR Policy, per TC labelled steered aggregate traffic counter (POL.BSID.TC)

This counter provides per Traffic Class (TC) breakdown of POL.BSID. It is defined as follows:

For each SR policy (P) and a given Traffic Class (TC), head-end node N MAY maintain counter(s) for the aggregate labelled traffic steered onto P.

2.5.5. Per-SR Policy, Per-Segment-List Aggregate traffic counter (POL.SL)

This counter is defined as:

For each SR policy (P) and a given Segment-List (SL), head-end node N SHOULD maintain counter(s) for the aggregate traffic steered onto the Segment-List (SL) of P.

2.5.6. Per-SR Policy, Per-Segment-List labelled steered aggregate traffic counter (POL.SL.BSID)

This counter is defined as:

For each SR policy (P) and a given Segment-List (SL), head-end node N MAY maintain counter(s) for the aggregate labelled traffic steered onto the Segment-List SL of P. Please note that labelled steered traffic refers to incoming packets with an active SID matching a local BSID of an SR policy at the head-end.

3. Security Considerations

This document does not define any new protocol extensions and does not impose any additional security challenges.

4. IANA Considerations

This document has no actions for IANA.

5. Acknowledgement

The authors like to thank Kris Michielsen for his valuable comments and suggestions.

6. Contributors

The following people have contributed to this document:

Ketan Talaulikar
Cisco Systems
Email: ketant@cisco.com

Siva Sivabalan
Cisco Systems
Email: msiva@cisco.com

Jose Liste
Cisco Systems
Email: jliste@cisco.com

Francois Clad
Cisco Systems
Email: fclad@cisco.com

Kamran Raza
Cisco Systems
Email: skraza@cisco.com

Shraddha Hegde
Juniper Networks
Email: shraddha@juniper.net

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

Rick Morton
Bell Canada
Email: rick.morton@bell.ca

Dirk Steinberg
Steinberg Consulting
Email: dws@steinbergnet.net

Bruno Decraene
Orange Business Services
Email: bruno.decraene@orange.com

Stephane Litkowski
Orange Business Services
Email: stephane.litkowski@orange.com

Luay Jalil
Verizon
Email: luay.jalil@verizon.com

7. References

7.1. Normative References

[I-D.filsfils-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Hegde, S., daniel.voyer@bell.ca, d., Lin, S., bogdanov@google.com, b., Krol, P., Horneffer, M., Steinberg, D., Decraene, B., Litkowski, S., Mattes, P., Ali, Z., Talaulikar, K., Liste, J., Clad, F., and K. Raza, "Segment Routing Policy Architecture", draft-filsfils-spring-segment-routing-policy-06 (work in progress), May 2018.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[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>.

7.2. Informative References

[I-D.ali-spring-sr-traffic-accounting] Ali, Z., Filsfils, C., Talaulikar, K., Sivabalan, S., Liste, J., Horneffer, M., Raszuk, R., Litkowski, S., Dawra, G., daniel.voyer@bell.ca, d., and R. Morton, "Traffic Accounting in Segment Routing Networks", draft-ali-spring-sr-traffic-accounting-01 (work in progress), May 2018.

[Traffic-Matrices] Schnitter, T-Systems, S. and M. Horneffer, T-Com, "Traffic Matrices for MPLS Networks with LDP Traffic Statistics, Proc. Networks2004, VDE-Verlag 2004", 2015.

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.

Email: cfilsfil@cisco.com

Zafar Ali (editor)
Cisco Systems, Inc.

Email: zali@cisco.com

Martin Horneffer
Deutsche Telekom

Email: martin.horneffer@telekom.de

Daniel Voyer
Bell Canada
671 de la gauchetiere W
Montreal, Quebec H3B 2M8
Canada

Email: daniel.voyer@bell.ca

Muhammad Durrani
Equinix

Email: mdurrani@equinix.com

Robert Raszuk
Bloomberg LP

Email: robert@raszuk.net

SPRING
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

C. Filsfils
P. Camarillo, Ed.
Cisco Systems, Inc.
J. Leddy
Comcast
D. Voyer
Bell Canada
S. Matsushima
SoftBank
Z. Li
Huawei Technologies
July 2, 2018

SRv6 Network Programming
draft-filsfils-spring-srv6-network-programming-05

Abstract

This document describes the SRv6 network programming concept and its most basic functions.

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 January 3, 2019.

Table of Contents

1. Introduction

Segment Routing leverages the source routing paradigm. An ingress node steers a packet through a ordered list of instructions, called segments. Each one of these instructions represents a function to be called at a specific location in the network. A function is locally defined on the node where it is executed and may range from simply moving forward in the segment list to any complex user-defined behavior. The network programming consists in combining segment routing functions, both simple and complex, to achieve a networking objective that goes beyond mere packet routing.

This document illustrates the SRv6 Network Programming concept and aims at standardizing the main segment routing functions to enable the creation of interoperable overlays with underlay optimization and service programming.

Familiarity with the Segment Routing Header [I-D.ietf-6man-segment-routing-header] is assumed.

2. Terminology

SRH is the abbreviation for the Segment Routing Header. We assume that the SRH may be present multiple times inside each packet.

NH is the abbreviation of the IPv6 next-header field.

NH=SRH means that the next-header field is 43 with routing type 4.

When there are multiple SRHs, they must follow each other: the next-header field of all SRH, except the last one, must be SRH.

The effective next-header (ENH) is the next-header field of the IP header when no SRH is present, or is the next-header field of the last SRH.

In this version of the document, we assume that there is no other extension header than the SRH. These will be lifted in future versions of the document.

SID: A Segment Identifier which represents a specific segment in segment routing domain. The SID type used in this document is IPv6 address (also referenced as SRv6 Segment or SRv6 SID).

A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SR path.

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

- IPv6 header with source and destination addresses respectively SA and DA and next-header is SRH

- SRH with SID list <S1, S2, S3> with SegmentsLeft = SL

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

- The payload of the packet is omitted.

SRH[SL] represents the SID pointed by the SL field in the first SRH. In our example, SRH[2] represents S1, SRH[1] represents S2 and SRH[0] represents S3.

FIB is the abbreviation for the forwarding table. A FIB lookup is a lookup in the forwarding table. When a packet is intercepted on a wire, it is possible that SRH[SL] is different from the DA.

3. SRv6 Segment

An SRv6 Segment is a 128-bit value. "SID" (abbreviation for Segment Identifier) is often used as a shorter reference for "SRv6 Segment".

An SRv6-capable node N maintains a "My Local SID Table". This table contains all the SRv6 segments explicitly instantiated at node N. N is the parent node for these SIDs.

A local SID of N can be an IPv6 address associated to a local interface of N but it is not mandatory. Nor is the My Local SID table populated by default with all IPv6 addresses defined on node N.

In most use-cases, a local SID will NOT be an address associated to a local interface of N.

A local SID of N could be routed to N but it does not have to be. Most often, it is routed to N via a shorter-mask prefix.

Let’s provide a classic illustration.

Node N is configured with a loopback0 interface address of C1::1/40 originated in its IGP. Node N is configured with two SIDs: C1::100 and C2::101.

The entry C1::1 is not defined explicitly as an SRv6 SID and hence does not appear in the "My Local SID Table". The entries C1::100 and C2::101 are defined explicitly as SRv6 SIDs and hence appear in the "My Local SID Table".

The network learns about a path to C1::/40 via the IGP and hence a packet destined to C1::100 would be routed up to N. The network does not learn about a path to C2::/40 via the IGP and hence a packet destined to C2::101 would not be routed up to N.

A packet could be steered to a non-routed SID C2::101 by using a SID list <...,C1::100,C2::101,...> where the non-routed SID is preceded by a routed SID to the same node. This is similar to the local vs global segments in SR-MPLS.

Every SRv6 SID instantiated has a specific instruction bound to it. This information is stored in the "My Local SID Table". The "My Local SID Table" has three main purposes:

- Define which local SIDs are explicitly instantiated

- Specify which instruction is bound to each of the instantiated SIDs

- Store the parameters associated with such instruction (i.e. OIF, NextHop,...)

We represent an SRv6 SID as LOC:FUNCT where LOC is the L most significant bits and FUNCT is the 128-L least significant bits. L is called the locator length and is flexible. Each operator is free to use the locator length it chooses. Most often the LOC part of the SID is routable and leads to the node which instantiates that SID.

Often, for simplicity of illustration, we will use a locator length of 64 bits. This is just an example. Implementations must not assume any a priori prefix length.

The FUNCT part of the SID is an opaque identification of a local function bound to the SID. Hence the name SRv6 Local SID.

A function may require additional arguments that would be placed in the rightmost-bits of the 128-bit space. In such case, the SRv6 SID will have the form LOC:FUNCT:ARGS.

These arguments may vary on a per-packet basis and may contain information related to the flow, service, or any other information required by the function associated to the SRv6 SID.

For this reason, the "My Local SID Table" matches on a per longest-prefix-match basis.

A node may receive a packet with an SRv6 SID in the DA without an SRH. In such case the packet should still be processed by the Segment Routing engine.

4. Functions associated with a SID

Each entry of the "My Local SID Table" indicates the function associated with the local SID and its parameters.

We define hereafter a set of well-known functions that can be associated with a SID.

End	Endpoint function The SRv6 instantiation of a prefix SID
End.X	Endpoint function with Layer-3 cross-connect The SRv6 instantiation of a Adj SID
End.T	Endpoint function with specific IPv6 table lookup
End.DX2	Endpoint with decapsulation and Layer-2 cross-connect L2VPN use-case
End.DX2V	Endpoint with decapsulation and VLAN L2 table lookup EVPN Flexible cross-connect use-cases
End.DT2U	Endpoint with decaps and unicast MAC L2 table lookup EVPN Bridging unicast use-cases
End.DT2M	Endpoint with decapsulation and L2 table flooding EVPN Bridging BUM use-cases with ESI filtering
End.DX6	Endpoint with decapsulation and IPv6 cross-connect IPv6 L3VPN use (equivalent of a per-CE VPN label)
End.DX4	Endpoint with decapsulation and IPv4 cross-connect IPv4 L3VPN use (equivalent of a per-CE VPN label)
End.DT6	Endpoint with decapsulation and IPv6 table lookup IPv6 L3VPN use (equivalent of a per-VRF VPN label)
End.DT4	Endpoint with decapsulation and IPv4 table lookup IPv4 L3VPN use (equivalent of a per-VRF VPN label)
End.DT46	Endpoint with decapsulation and IP table lookup IP L3VPN use (equivalent of a per-VRF VPN label)
End.B6	Endpoint bound to an SRv6 policy SRv6 instantiation of a Binding SID
End.B6.Encaps	Endpoint bound to an SRv6 encapsulation Policy SRv6 instantiation of a Binding SID
End.BM	Endpoint bound to an SR-MPLS Policy SRv6/SR-MPLS instantiation of a Binding SID
End.S	Endpoint in search of a target in table T

The list is not exhaustive. In practice, any function can be attached to a local SID: e.g. a node N can bind a SID to a local VM or container which can apply any complex function on the packet.

We call N the node who has an explicitly instantiated SID S and we detail the function that N binds to S.

At the end of this section we also present some flavours of these well-known functions.

4.1. End: Endpoint

The Endpoint function ("End" for short) is the most basic function.

When N receives a packet whose IPv6 DA is S and S is a local End SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      decrement SL
3.      update the IPv6 DA with SRH[SL]
4.      FIB lookup on the updated DA                          ;; Ref1
5.      forward accordingly to the matched entry              ;; Ref2
6.   ELSE
7.      drop the packet                                       ;; Ref3
```

Ref1: The End function performs the FIB lookup in the forwarding table associated to the ingress interface

Ref2: If the FIB lookup matches a multicast state, then the related RPF check must be considered successful

Ref3: a local SID could be bound to a function which authorizes the decapsulation of an outer header (e.g. IPinIP) or the punting of the packet to TCP, UDP or any other protocol. This however needs to be explicitly defined in the function bound to the local SID. By default, a local SID bound to the well-known function "End" only allows the punting to OAM protocols and neither allows the decapsulation of an outer header nor the cleanup of an SRH. As a consequence, an End SID cannot be the last SID of an SRH and cannot be the DA of a packet without SRH.

This is the SRv6 instantiation of a Prefix SID [I-D.ietf-spring-segment-routing].

4.2. End.X: Endpoint with Layer-3 cross-connect

The "Endpoint with cross-connect to an array of layer-3 adjacencies" function (End.X for short) is a variant of the End function.

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

```
1.   IF NH=SRH and SL > 0
2.      decrement SL
3.      update the IPv6 DA with SRH[SL]
4.      forward to layer-3 adjacency bound to the SID S       ;; Ref1
5.   ELSE
6.      drop the packet                                       ;; Ref2
```

Ref1: If an array of adjacencies is bound to the End.X SID, then one entry of the array is selected based on a hash of the packet's header.

Ref2: An End.X function only allows punting to OAM and does not allow decaps. An End.X SID cannot be the last SID of an SRH and cannot be the DA of a packet without SRH.

The End.X function is required to express any traffic-engineering policy.

This is the SRv6 instantiation of an Adjacency SID [I-D.ietf-spring-segment-routing].

If a node N has 30 outgoing interfaces to 30 neighbors, usually the operator would explicitly instantiate 30 End.X SIDs at N: one per layer-3 adjacency to a neighbor. Potentially, more End.X could be explicitly defined (groups of layer-3 adjacencies to the same neighbor or to different neighbors).

Note that with SR-MPLS, an AdjSID is typically preceded by a PrefixSID. This is unlikely in SRv6 as most likely an End.X SID is globally routed to N.

Note that if N has an outgoing interface bundle I to a neighbor Q made of 10 member links, N may allocate up to 11 End.X local SIDs for that bundle: one for the bundle itself and then up to one for each member link. This is the equivalent of the L2-Link Adj SID in SR-MPLS [I-D.ietf-isis-l2bundles].

4.3. End.T: Endpoint with specific IPv6 table lookup

The "Endpoint with specific IPv6 table lookup" function (End.T for short) is a variant of the End function.

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

```
1.   IF NH=SRH and SL > 0                                    ;; Ref1
2.      decrement SL
3.      update the IPv6 DA with SRH[SL]
4.      lookup the next segment in IPv6 table T associated with the SID
5.      forward via the matched table entry
6.   ELSE
7.      drop the packet
```

Ref1: The End.T SID must not be the last SID

The End.T is used for multi-table operation in the core.

4.4. End.DX2: Endpoint with decapsulation and Layer-2 cross-connect

The "Endpoint with decapsulation and Layer-2 cross-connect to OIF" function (End.DX2 for short) is a variant of the endpoint function.

When N receives a packet destined to S and S is a local End.DX2 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH  = 59                                     ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      forward the resulting frame via OIF associated to the SID
6.   ELSE
7.      drop the packet
```

Ref1: An End.DX2 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: We conveniently reuse the next-header value 59 allocated to IPv6 No Next Header [RFC8200]. When the SID corresponds to function End.DX2 and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

An End.DX2 function could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

One of the applications of the End.DX2 function is the L2VPN use-case.

4.5. End.DX2V: Endpoint with decapsulation and VLAN L2 table lookup

The "Endpoint with decapsulation and specific VLAN L2 table lookup" function (End.DX2V for short) is a variant of the endpoint function.

When N receives a packet destined to S and S is a local End.DX2V SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                        ;; Ref1
3.   ELSE IF ENH  = 59                                         ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      lookup the exposed inner VLANs in L2 table T
6.      forward via the matched table entry
7.   ELSE
8.      drop the packet
```

Ref1: An End.DX2V SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: We conveniently reuse the next-header value 59 allocated to IPv6 No Next Header [RFC8200]. When the SID corresponds to function End.DX2V and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

An End.DX2V function could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

The End.DX2V is used for EVPN Flexible cross-connect use-cases.

4.6. End.DT2U: Endpoint with decapsulation and unicast MAC L2 table lookup

The "Endpoint with decapsulation and specific unicast MAC L2 table lookup" function (End.DT2U for short) is a variant of the endpoint function.

When N receives a packet destined to S and S is a local End.DT2U SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                        ;; Ref1
3.   ELSE IF ENH  = 59                                         ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      learn the exposed inner MAC SA in L2 table T           ;; Ref3
6.      lookup the exposed inner MAC DA in L2 table T
7.      forward via the matched T entry else to all L2OIF in T
8.   ELSE
9.      drop the packet
```

Ref1: An End.DT2U SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: We conveniently reuse the next-header value 59 allocated to IPv6 No Next Header [RFC8200]. When the SID corresponds to function

End.DT2U and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

Ref3: In EVPN, the learning of the exposed inner MAC SA is done via control plane.

The End.DT2U is used for EVPN Bridging unicast use cases.

4.7. End.DT2M: Endpoint with decapsulation and L2 table flooding

The "Endpoint with decapsulation and specific L2 table flooding" function (End.DT2M for short) is a variant of the endpoint function.

This function may take an argument: "Arg.FE2". It is an argument specific to EVPN ESI filtering. It is used to exclude a specific OIF from L2 table T flooding.

When N receives a packet destined to S and S is a local End.DT2M SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                        ;; Ref1
3.   ELSE IF ENH  = 59                                        ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      learn the exposed inner MAC SA in L2 table T           ;; Ref3
6.      forward on all L2OIF excluding the one specified in Arg.FE2
7.   ELSE
8.      drop the packet
```

Ref1: An End.DT2M SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: We conveniently reuse the next-header value 59 allocated to IPv6 No Next Header [RFC8200]. When the SID corresponds to function End.DT2M and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

Ref3: In EVPN, the learning of the exposed inner MAC SA is done via control plane

The End.DT2M is used for EVPN Bridging BUM use case with ESI filtering capability.

4.8. End.DX6: Endpoint with decapsulation and IPv6 cross-connect

The "Endpoint with decapsulation and cross-connect to an array of IPv6 adjacencies" function (End.DX6 for short) is a variant of the End and End.X functions.

When N receives a packet destined to S and S is a local End.DX6 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH = 41                                      ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      forward to layer-3 adjacency bound to the SID S    ;; Ref3
6.   ELSE
7.      drop the packet
```

Ref1: The End.DX6 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers

Ref3: Selected based on a hash of the packet’s header (at least SA, DA, Flow Label)

One of the applications of the End.DX6 function is the L3VPN use-case where a FIB lookup in a specific tenant table at the egress PE is not required. This would be equivalent to the per-CE VPN label in MPLS[RFC4364].

4.9. End.DX4: Endpoint with decapsulation and IPv4 cross-connect

The "Endpoint with decapsulation and cross-connect to an array of IPv4 adjacencies" function (End.DX4 for short) is a variant of the End and End.X functions.

When N receives a packet destined to S and S is a local End.DX4 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH  =  4                                     ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      forward to layer-3 adjacency bound to the SID S    ;; Ref3
6.   ELSE
7.      drop the packet
```

Ref1: The End.DX4 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers

Ref3: Selected based on a hash of the packet's header (at least SA, DA, Flow Label)

One of the applications of the End.DX4 function is the L3VPN use-case where a FIB lookup in a specific tenant table at the egress PE is not required. This would be equivalent to the per-CE VPN label in MPLS[RFC4364].

4.10. End.DT6: Endpoint with decapsulation and specific IPv6 table lookup

The "Endpoint with decapsulation and specific IPv6 table lookup" function (End.DT6 for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.DT6 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH = 41                                      ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      lookup the exposed inner IPv6 DA in IPv6 table T
6.      forward via the matched table entry
7.   ELSE
8.      drop the packet
```

Ref1: the End.DT6 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers

One of the applications of the End.DT6 function is the L3VPN use-case where a FIB lookup in a specific tenant table at the egress PE is required. This would be equivalent to the per-VRF VPN label in MPLS[RFC4364].

Note that an End.DT6 may be defined for the main IPv6 table in which case and End.DT6 supports the equivalent of an IPv6inIPv6 decaps (without VPN/tenant implication).

4.11. End.DT4: Endpoint with decapsulation and specific IPv4 table lookup

The "Endpoint with decapsulation and specific IPv4 table lookup" function (End.DT4 for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.DT4 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                        ;; Ref1
3.   ELSE IF ENH = 4                                          ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      lookup the exposed inner IPv4 DA in IPv4 table T
6.      forward via the matched table entry
7.   ELSE
8.      drop the packet
```

Ref1: the End.DT4 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers

One of the applications of the End.DT4 is the L3VPN use-case where a FIB lookup in a specific tenant table at the egress PE is required. This would be equivalent to the per-VRF VPN label in MPLS[RFC4364].

Note that an End.DT4 may be defined for the main IPv4 table in which case and End.DT4 supports the equivalent of an IPv4inIPv6 decaps (without VPN/tenant implication).

4.12. End.DT46: Endpoint with decapsulation and specific IP table lookup

The "Endpoint with decapsulation and specific IP table lookup" function (End.DT46 for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.DT46 SID, N does:

```
 1.   IF NH=SRH and SL > 0
 2.      drop the packet                                        ;; Ref1
 3.   ELSE IF ENH  =  4                                        ;; Ref2
 4.      pop the (outer) IPv6 header and its extension headers
 5.      lookup the exposed inner IPv4 DA in IPv4 table T
 6.      forward via the matched table entry
 7.   ELSE IF ENH = 41                                         ;; Ref2
 8.      pop the (outer) IPv6 header and its extension headers
 9.      lookup the exposed inner IPv6 DA in IPv6 table T
10.      forward via the matched table entry
11.   ELSE
12.      drop the packet
```

Ref1: the End.DT46 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 4 and 41 refer to IPv4 and IPv6 encapsulation respectively as defined by IANA allocation for Internet Protocol Numbers

One of the applications of the End.DT46 is the L3VPN use-case where a FIB lookup in a specific tenant table at the egress PE is required. This would be equivalent to the per-VRF VPN label in MPLS[RFC4364].

Note that an End.DT46 may be defined for the main IP table in which case and End.DT46 supports the equivalent of an IPinIPv6 decaps (without VPN/tenant implication).

4.13. End.B6: Endpoint bound to an SRv6 policy

The "Endpoint bound to an SRv6 Policy" is a variant of the End function.

When N receives a packet destined to S and S is a local End.B6 SID, N does:

```
1.   IF NH=SRH and SL > 0                                    ;; Ref1
2.      do not decrement SL nor update the IPv6 DA with SRH[SL]
3.      insert a new SRH                                     ;; Ref2
4.      set the IPv6 DA to the first segment of the SRv6 Policy
5.      forward according to the first segment of the SRv6 Policy
6.   ELSE
7.      drop the packet
```

Ref1: An End.B6 SID, by definition, is never the last SID.

Ref2: In case that an SRH already exists, the new SRH is inserted in between the IPv6 header and the received SRH

Note: Instead of the term "insert", "push" may also be used.

The End.B6 function is required to express scalable traffic-engineering policies across multiple domains. This is the SRv6 instantiation of a Binding SID [I-D.ietf-spring-segment-routing].

4.14. End.B6.Red: Endpoint bound to an SRv6 reduced policy

This is an optimization of the End.B6 function.

End.B6.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the pushed SRH. In this way, the

first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

4.15. End.B6.Encaps: Endpoint bound to an SRv6 encapsulation policy

This is a variation of the End.B6 behavior where the SRv6 Policy also includes an IPv6 Source Address A.

When N receives a packet destined to S and S is a local End.B6.Encaps SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      decrement SL and update the IPv6 DA with SRH[SL]
3.      push an outer IPv6 header with its own SRH
4.      set the outer IPv6 SA to A
5.      set the outer IPv6 DA to the first segment of the SRv6 Policy
6.      forward according to the first segment of the SRv6 Policy
7.   ELSE
8.      drop the packet
```

Instead of simply inserting an SRH with the policy (End.B6), this behavior also adds an outer IPv6 header. The source address defined for the outer header does not have to be a local SID of the node.

4.16. End.B6.Encaps.Red: Endpoint bound to an SRv6 reduced encapsulation policy

This is an optimization of the End.B6.Encaps function.

End.B6.Encaps.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the pushed SRH. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

4.17. End.BM: Endpoint bound to an SR-MPLS policy

The "Endpoint bound to an SR-MPLS Policy" is a variant of the End.B6 function.

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

```
1.   IF NH=SRH and SL > 0                                    ;; Ref1
2.      decrement SL and update the IPv6 DA with SRH[SL]
3.      push an MPLS label stack <L1, L2, L3> on the received packet
4.      forward according to L1
5.   ELSE
6.      drop the packet
```

Ref1: an End.BM SID, by definition, is never the last SID.

The End.BM function is required to express scalable traffic-engineering policies across multiple domains where some domains support the MPLS instantiation of Segment Routing.

This is an SRv6 instantiation of a SR-MPLS Binding SID [I-D.ietf-spring-segment-routing].

4.18. End.S: Endpoint in search of a target in table T

The "Endpoint in search of a target in Table T" function (End.S for short) is a variant of the End function.

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

```
1.   IF NH=SRH and SL = 0                                    ;; Ref1
2.      drop the packet
3.   ELSE IF match(last SID) in specified table T
4.      forward accordingly
5.   ELSE
6.      apply the End behavior
```

Ref1: By definition, an End.S SID cannot be the last SID, as the last SID is the targeted object.

The End.S function is required in information-centric networking (ICN) use-cases where the last SID in the SRv6 SID list represents a targeted object. If the identification of the object would require more than 128 bits, then obvious customization of the End.S function may either use multiple SIDs or a TLV of the SR header to encode the searched object ID.

4.19. SR-aware application

Generally, any SR-aware application can be bound to an SRv6 SID. This application could represent anything from a small piece of code focused on topological/tenant function to a much larger process focusing on higher-level applications (e.g. video compression, transcoding etc.).

The ways in which an SR-aware application can binds itself on a local SID depends on the operating system. Let us consider an SR-aware application running on a Linux operating system. A possible approach is to associate an SRv6 SID to a target (virtual) interface, so that packets with IP DA corresponding to the SID will be sent to the target interface. In this approach, the SR-aware application can simply listen to all packets received on the interface.

A different approach for the SR-aware app is to listen to packets received with a specific SRv6 SID as IPv6 DA on a given transport port (i.e. corresponding to a TCP or UDP socket). In this case, the app can read the SRH information with a getsockopt Linux system call and can set the SRH information to be added to the outgoing packets with a setsocksopt system call.

4.20. Non SR-aware application

[I-D.xuclad-spring-sr-service-programming] defines a set of additional functions in order to enable non SR-aware applications to be associated with a SRv6 Local SID.

4.21. Flavours

We present the PSP and USP variants of the functions End, End.X and End.T. For each of these functions these variants can be enabled or disabled either individually or together.

4.21.1. PSP: Penultimate Segment Pop of the SRH

After the instruction 'update the IPv6 DA with SRH[SL]' is executed, the following instructions must be added:

```
1.   IF updated SL = 0 & PSP is TRUE & O-bit = 0 & A-bit = 0    ;; Ref1
2.      pop the top SRH                                         ;; Ref2
```

Ref1: If the SRH.Flags.O-bit or SRH.Flags.A-bit is set, PSP of the SRH is disabled. Section 6.1 specifies the pseudocode needed to process the SRH.Flags.O-bit.

Ref2: The received SRH had SL=1. When the last SID is written in the DA, the End, End.X and End.T functions with the PSP flavour pop the first (top-most) SRH. Subsequent stacked SRH's may be present but are not processed as part of the function.

4.21.2. USP: Ultimate Segment Pop of the SRH

We insert at the beginning of the pseudo-code the following instructions:

```
1.   IF NH=SRH & SL = 0 & USP=TRUE                              ;; Ref1
2.      pop the top SRH
3.      restart the function processing on the modified packet  ;; Ref2
```

Ref1: The next header is an SRH header

Ref2: Typically SL of the exposed SRH is > 0 and hence the restarting of the complete function would lead to decrement SL, update the IPv6 DA with SRH[SL], FIB lookup on updated DA and forward accordingly to the matched entry.

5. Transit behaviors

We define hereafter the set of basic transit behaviors.

T	Transit behavior
T.Insert	Transit behavior with insertion of an SRv6 policy
T.Insert.Red	Transit behavior with reduced insert of an SRv6 policy
T.Encaps	Transit behavior with encapsulation in an SRv6 policy
T.Encaps.Red	Transit behavior with reduced encaps in an SRv6 policy
T.Encaps.L2	T.Encaps behavior of the received L2 frame
T.Encaps.L2.Red	Transit with reduce encaps of received L2 frame

This list can be expanded in case any new functionality requires it.

5.1. T: Transit behavior

As per [RFC8200], if a node N receives a packet (A, S2)(S3, S2, S1; SL=2) and S2 is neither a local address nor a local SID of N then N forwards the packet without inspecting the SRH.

This means that N treats the following two packets with the same performance:

- (A, S2)

- (A, S2)(S3, S2, S1; SL=2)

A transit node does not need to count by default the amount of transit traffic with an SRH extension header. This accounting might be enabled as an optional behavior leveraging SEC4 behavior described later in this document.Section 7.4

A transit node MUST include the outer flow label in its ECMP hash[RFC6437].

5.2. T.Insert: Transit with insertion of an SRv6 Policy

Node N receives two packets P1=(A, B2) and P2=(A,B2)(B3, B2, B1; SL=1). B2 is neither a local address nor SID of N.

N steers the transit packets P1 and P2 into an SRv6 Policy with one SID list <S1, S2, S3>.

The "T.Insert" transit insertion behavior is defined as follows:

```
1.  insert the SRH (B2, S3, S2, S1; SL=3)            ;; Ref1, Ref1bis
2.  set the IPv6 DA = S1
3.  forward along the shortest path to S1
```

Ref1: The received IPv6 DA is placed as last SID of the inserted SRH.

Ref1bis: The SRH is inserted before any other IPv6 Routing Extension Header.

After the T.Insert behavior, P1 and P2 respectively look like:

- (A, S1) (B2, S3, S2, S1; SL=3)

- (A, S1) (B2, S3, S2, S1; SL=3) (B3, B2, B1; SL=1)

5.3. T.Insert.Red: Transit with reduced insertion of an SRv6 Policy

The T.Insert.Red behavior is an optimization of the T.Insert behavior. It is defined as follows:

```
1.  insert the SRH (B2, S3, S2; SL=3)
2.  set the IPv6 DA = S1
3.  forward along the shortest path to S1
```

T.Insert.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the pushed SRH. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

After the T.Insert.Red behavior, P1 and P2 respectively look like:

- (A, S1) (B2, S3, S2; SL=3)

- (A, S1) (B2, S3, S2; SL=3) (B3, B2, B1; SL=1)

5.4. T.Encaps: Transit with encapsulation in an SRv6 Policy

Node N receives two packets P1=(A, B2) and P2=(A,B2)(B3, B2, B1; SL=1). B2 is neither a local address nor SID of N.

N steers the transit packets P1 and P2 into an SR Encapsulation Policy with a Source Address T and a Segment list <S1, S2, S3>.

The T.Encaps transit encapsulation behavior is defined as follows:

```
1.  push an IPv6 header with its own SRH (S3, S2, S1; SL=2)
2.  set outer IPv6 SA = T and outer IPv6 DA = S1
3.  set outer payload length, traffic class and flow label     ;; Ref1
4.  update the Next-Header value                               ;; Ref1
5.  decrement inner Hop Limit or TTL                           ;; Ref1
6.  forward along the shortest path to S1
```

After the T.Encaps behavior, P1 and P2 respectively look like:

- (T, S1) (S3, S2, S1; SL=2) (A, B2)

- (T, S1) (S3, S2, S1; SL=2) (A, B2) (B3, B2, B1; SL=1)

The T.Encaps behavior is valid for any kind of Layer-3 traffic. This behavior is commonly used for L3VPN with IPv4 and IPv6 deployements.

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

Ref 1: As described in [RFC2473] (Generic Packet Tunneling in IPv6 Specification)

5.5. T.Encaps.Red: Transit with reduce encaps in an SRv6 Policy

The T.Encaps.Red behavior is an optimization of the T.Encaps behavior. It is defined as follows:

```
1.  push an IPv6 header with its own SRH (S3, S2; SL=2)
2.  set outer IPv6 SA = T and outer IPv6 DA = S1
3.  set outer payload length, traffic class and flow label     ;; Ref1
4.  update the Next-Header value                               ;; Ref1
5.  decrement inner Hop Limit or TTL                           ;; Ref1
6.  forward along the shortest path to S1
```

Ref 1: As described in [RFC2473] (Generic Packet Tunneling in IPv6 Specification)

T.Encaps.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the SRH of the pushed IPv6 packet. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

After the T.Encaps.Red behavior, P1 and P2 respectively look like:

- (T, S1) (S3, S2; SL=2) (A, B2)

- (T, S1) (S3, S2; SL=2) (A, B2) (B3, B2, B1; SL=1)

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

5.6. T.Encaps.L2: Transit with encapsulation of L2 frames

While T.Encaps encapsulates the received IP packet, T.Encaps.L2 encapsulates the received L2 frame (i.e. the received ethernet header and its optional VLAN header is in the payload of the outer packet).

If the outer header is pushed without SRH then the DA must be a SID of type End.DX2, End.DX2V, End.DT2U or End.DT2M and the next-header must be 59 (IPv6 NoNextHeader). The received Ethernet frame follows the IPv6 header and its extension headers.

Else, if the outer header is pushed with an SRH, then the last SID of the SRH must be of type End.DX2, End.DX2V, End.DT2U or End.DT2M and the next-header of the SRH must be 59 (IPv6 NoNextHeader). The received Ethernet frame follows the IPv6 header and its extension headers.

5.7. T.Encaps.L2.Red: Transit with reduce encaps of L2 frames in an SRv6 Policy

The T.Encaps.L2.Red behavior is an optimization of the T.Encaps.L2 behavior.

T.Encaps.L2.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the SRH of the pushed IPv6 packet. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

6. Operation

6.1. Counters

Any SRv6 capable node SHOULD implement the following set of combined counters (packets and bytes):

- CNT1: Per entry of the "My Local SID Table", traffic that matched that SID and was processed correctly.

- CNT2: Per SRv6 Policy, traffic steered into it and processed correctly.

Furthermore, an SRv6 capable node maintains an aggregate counter CNT3 tracking the IPv6 traffic that was received with a destination address matching a local interface address that is not a local SID and the next-header is SRH with SL>0. We remind that this traffic is dropped as an interface address is not a local SID by default. A SID must be explicitly instantiated.

6.2. Flow-based hash computation

When a flow-based selection within a set needs to be performed, the source address, the destination address and the flow-label MUST be included in the flow-based hash.

This occurs when the destination address is updated, a FIB lookup is performed and multiple ECMP paths exist to the updated destination address.

This occurs when End.X is bound to an array of adjacencies.

This occurs when the packet is steered in an SR policy whose selected path has multiple SID lists [I-D.filsfils-spring-segment-routing-policy].

6.3. OAM

[I-D.ali-spring-srv6-oam] defines the OAM behavior for SRv6. This includes the definition of the SRH Flag 'O-bit', as well as additional OAM Endpoint functions.

7. Basic security for intra-domain deployment

We use the following terminology:

An internal node is a node part of the domain of trust.

A border router is an internal node at the edge of the domain of trust.

An external interface is an interface of a border router towards another domain.

An internal interface is an interface entirely within the domain of trust.

The internal address space is the IP address block dedicated to internal interfaces.

An internal SID is a SID instantiated on an internal node.

The internal SID space is the IP address block dedicated to internal SIDs.

External traffic is traffic received from an external interface to the domain of trust.

Internal traffic is traffic that originates and ends within the domain of trust.

The purpose of this section is to document how a domain of trust can operate SRv6-based services for internal traffic while preventing any external traffic from accessing the internal SRv6-based services.

It is expected that future documents will detail enhanced security mechanisms for SRv6 (e.g. how to allow external traffic to leverage internal SRv6 services).

7.1. SEC 1

An SRv6 router MUST support an ACL on the external interface that drops any traffic with SA or DA in the internal SID space.

A provider would generally do this for its internal address space to prevent access to internal addresses and in order to prevent spoofing. The technique is extended to the local SID space.

The typical counters of an ACL are expected.

7.2. SEC 2

An SRv6 router MUST support an ACL with the following behavior:

```
1.   IF (DA == LocalSID) && (SA != internal address or SID space)
2.      drop
```

This prevents access to local SIDs from outside the operator's infrastructure. Note that this ACL may not be enabled in all cases. For example, specific SIDs can be used to provide resources to devices that are outside of the operator's infrastructure.

When an SID is in the form of LOC:FUNCT:ARGS the DA match should be implemented as a prefix match covering the argument space of the specific SID i.s.o. a host route.

The typical counters of an ACL are expected.

7.3. SEC 3

As per the End definition, an SRv6 router MUST only implement the End behavior on a local IPv6 address if that address has been explicitly enabled as a segment.

This address may or may not be associated with an interface. This address may or may not be routed. The only thing that matters is that the local SID must be explicitly instantiated and explicitly bound to a function (the default function is the End function).

7.4. SEC 4

An SRv6 router should support Unicast-RPF on source address on external interface.

This is a generic provider technique applied to the internal address space. It is extended to the internal SID space.

The typical counters to validate such filtering are expected.

8. Control Plane

In an SDN environment, one expects the controller to explicitly provision the SIDs and/or discover them as part of a service discovery function. Applications residing on top of the controller could then discover the required SIDs and combine them to form a distributed network program.

The concept of "SRv6 network programming" refers to the capability for an application to encode any complex program as a set of individual functions distributed through the network. Some functions relate to underlay SLA others to overlay/tenant, others to complex applications residing in VM and containers.

The specification of the SRv6 control-plane is outside the scope of this document.

We limit ourselves to a few important observations.

8.1. IGP

The End and End.X SIDs express topological functions and hence are expected to be signaled in the IGP together with the flavours PSP and USP [I-D.bashandy-isis-srv6-extensions].

The presence of SIDs in the IGP do not imply any routing semantics to the addresses represented by these SIDs. The routing reachability to an IPv6 address is solely governed by the classic, non-SID-related, IGP information. Routing is not governed neither influenced in any way by a SID advertisement in the IGP.

These two SIDs provide important topological functions for the IGP to build FRR/TI-LFA solution and for TE processes relying on IGP LSDB to build SR policies.

8.2. BGP-LS

BGP-LS is expected to be the key service discovery protocol. Every node is expected to advertise via BGP-LS its SRv6 capabilities (e.g. how many SIDs in can insert as part of an T.Insert behavior) and any locally instantiated SID[I-D.ietf-idr-bgp-ls-segment-routing-ext][I-D.ietf-idr-te-lsp-distribution].

8.3. BGP IP/VPN

The End.DX4, End.DX6, End.DT4, End.DT6, End.DT46 and End.DX2 SIDs are expected to be signaled in BGP[I-D.dawra-idr-srv6-vpn].

8.4. Summary

The following table summarizes which SID would be signaled in which signaling protocol.

	IGP	BGP-LS	BGP IP/VPN
End (PSP, USP)	X	X	
End.X (PSP, USP)	X	X	
End.T (PSP, USP)	X	X	
End.DX2		X	X
End.DX2V		X	X
End.DT2U		X	X
End.DT2M		X	X
End.DX6	X	X	X
End.DX4		X	X
End.DT6	X	X	X
End.DT4		X	X
End.DT46		X	X
End.B6		X	
End.B6.Encaps		X	
End.B6.BM		X	
End.S		X	

Table 1: SRv6 LocalSID signaling

The following table summarizes which transit capability would be signaled in which signaling protocol.

	IGP	BGP-LS	BGP IP/VPN
T		X	
T.Insert		X	
T.Encaps		X	
T.Encaps.L2		X	

Table 2: SRv6 transit behaviors signaling

The previous table describes generic capabilities. It does not describe specific instantiated SID.

For example, a BGP-LS advertisement of the T capability of node N would indicate that node N supports the basic transit behavior. The T.Insert behavior would describe the capability of node N to instantiation a T.Insert behavior, specifically it would describe how many SIDs could be inserted by N without significant performance degradation. Same for T.Encaps (the number potentially lower as the overhead of the additional outer IP header is accounted).

The reader should also remember that any specific instantiated SR policy (via T.Insert or T.Encaps) is always assigned a Binding SID. They should remember that BSIDs are advertised in BGP-LS as shown in Table 1. Hence, it is normal that Table 2 only focuses on the generic capabilities related to T.Insert and T.Encaps as Table 1 advertises the specific instantiated BSID properties.

9. Illustration

We introduce a simplified SID allocation technique to ease the reading of the text. We document the reference diagram. We then illustrate the network programming concept through different use-cases. These use-cases have been thought to allow straightforward combination between each other.

9.1. Simplified SID allocation

To simplify the illustration, we assume:

A::/4 is dedicated to the internal SRv6 SID space

B::/4 is dedicated to the internal address space

We assume a location expressed in 48 bits and a function expressed in 80 bits

Node k has a classic IPv6 loopback address Bk::/128 which is advertised in the IGP

Node k has Ak::/48 for its local SID space. Its SIDs will be explicitly allocated from that block

Node k advertises Ak::/48 in its IGP

Function 0:0:0:0:1 (function 1, for short) represents the End function with PSP support

Function 0:0:0:0:C2 (function C2, for short) represents the End.X function towards neighbor 2

Each node K has:

An explicit SID instantiation Ak::1/128 bound to an End function with additional support for PSP

An explicit SID instantiation Ak::Cj/128 bound to an End.X function to neighbor J with additional support for PSP

9.2. Reference diagram

Let us assume the following topology where all the links have IGP metric 10 except the link 23 which is 100.

Nodes A, 1 to 8 and B are considered within the network domain while nodes CE-A and CE-B are outside the domain.

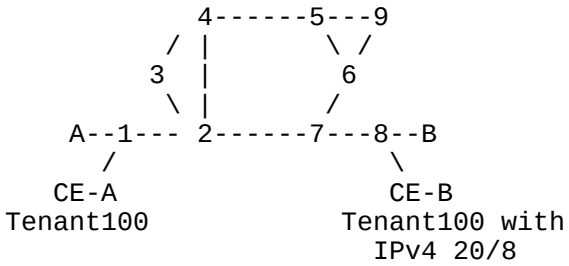


Figure 1: Reference topology

9.3. Basic security

Any edge node such as 1 would be configured with an ACL on any of its external interface (e.g. from CE-A) which drops any traffic with SA or DA in A::/4. See SEC 1 (Section 7.1).

Any core node such as 6 could be configured with an ACL with the SEC2 (Section 7.2) behavior "IF (DA == LocalSID) && (SA is not in A::/4 or B::/4) THEN drop".

SEC 3 (Section 7.3) protection is a default property of SRv6. A SID must be explicitly instantiated. In our illustration, the only available SIDs are those explicitly instantiated.

Any edge node such as 1 would be configured with Unicast-RPF on source address on external interface (e.g. from CE-A). See SEC 4 (Section 7.4).

9.4. SR-IPVPN

Let us illustrate the SR-IPVPN use-case applied to IPv4.

Nodes 1 and 8 are configured with a tenant 100, each respectively connected to CE-A and CE-B.

Node 8 is configured with a local SID A8::D100 of function End.DT4 bound to tenant IPv4 table 100.

Via BGP signaling or an SDN-based controller, Node 1's tenant-100 IPv4 table is programmed with an IPv4 SR-VPN route 20/8 via SRv6 policy <A8::D100>.

When 1 receives a packet P from CE-A destined to 20.20.20.20, 1 looks up P in its tenant-100 IPv4 table and finds an SR-VPN entry 20/8 via SRv6 policy <A8::D100>. As a consequence, 1 pushes an outer IPv6 header with SA=A1::0, DA=A8::D100 and NH=4. 1 then forwards the resulting packet on the shortest path to A8::/40.

When 8 receives the packet, 8 matches the DA in its My LocalSID table, finds the bound function End.DT4(100) and confirms NH=4. As a result, 8 decaps the outer header, looks up the inner IPv4 DA in tenant-100 IPv4 table, and forward the (inner) IPv4 packet towards CE-B.

The reader can easily infer all the other SR-IPVPN IP instantiations:

Route at ingress PE(1)	SR-VPN Egress SID of egress PE(8)
IPv4 tenant route with egress tenant table lookup	End.DT4 function bound to IPv4-tenant-100 table
IPv4 tenant route without egress tenant table lookup	End.DX4 function bound to CE-B (IPv4)
IPv6 tenant route with egress tenant table lookup	End.DT6 function bound to IPv6-tenant-100 table
IPv6 tenant route without egress tenant table lookup	End.DX6 function bound to CE-B (IPv6)

9.5. SR-Ethernet-VPWS

Let us illustrate the SR-Ethernet-VPWS use-case.

Node 1 is configured with an ethernet VPWS service:

- Local attachment circuit: Ethernet interface from CE-A

- Local End.DX2 bound to the local attachment circuit: A1::DC2A

- Remote End.DX2 SID: A8::DC2B

Node 8 is configured with an ethernet VPWS service:

- Local attachment circuit: Ethernet interface from CE-B

- Local End.DX2 bound to the local attachment circuit: A8::DC2B

- Remote End.DX2 SID: A1::DC2A

These configurations can either be programmed by an SDN controller or partially derived from a BGP-based signaling and discovery service.

When 1 receives a frame F from CE-A, 1 pushes an outer IPv6 header with SA=A1::0, DA=A8::DC2B and NH=59. Note that no additional header is pushed. 1 then forwards the resulting packet on the shortest path to A8::/40.

When 8 receives the packet, 8 matches the DA in its My LocalSID table and finds the bound function End.DX2. After confirming that the next-header=59, 8 decaps the outer IPv6 header and forwards the inner Ethernet frame towards CE-B.

The reader can easily infer the Ethernet VPWS use-case:

Route at ingress PE(1)	SR-VPN Egress SID of egress PE(8)
Ethernet VPWS	End.DX2 function bound to CE-B (Ethernet)

9.6. SR-EVPN-FXC

Let us illustrate the SR-EVPN-FXC use-case (Flexible cross-connect service).

Node 1 is configured with an EVPN-FXC service:

- Local attachment circuit: Ethernet interface from CE1-A over VLAN 100

- Local attachment circuit: Ethernet interface from CE2-A over VLAN 200

- Local End.DX2 bound to the local attachment circuit: A1::DC2A

- Remote End.DX2 SID: A8::DC2B

Node 8 is configured with an EVPN-FXC service:

- Local attachment circuit: Ethernet interface from CE1-B over VLAN 100

- Local attachment circuit: Ethernet interface from CE2-B over VLAN 200

- Local End.DX2 bound to the local attachment circuit: A8::DC2B

- Remote End.DX2 SID: A1::DC2A

These configurations can either be programmed by an SDN controller or derived from a BGP-based EVPN-FXC service. EVPN route Type-1 is used for that purpose.

When node 1 receives a frame F from CE-A, it pushes an outer IPv6 header with SA=A1::0, DA=A8::DC2B and NH=59. Note that no additional header is pushed. Node 1 then forwards the resulting packet on the shortest path to A8::/40.

When node 8 receives the packet, it matches the IP DA in its My LocalSID table and finds the bound function End.DX2V. After confirming that the next-header=59, node 8 decaps the outer IPv6 header, performs a VLAN loopkup in table T1 and forwards the inner Ethernet frame to matching interface e.g. for VLAN 100, packet is forwarded to CE1-B and for VLAN 200, frame is forwarded to CE2-B.

The reader can easily infer the Ethernet FXC use-case:

Route at ingress PE (1)	SR-VPN Egress SID of egress PE (8)
EVPN-FXC	End.DX2V function bound to CE1-B / CE2-B (Ethernet)

9.7. SR-EVPN

There are few use cases to illustrate under SR-EVPN: bridging (unicast and multicast), multi-homing ESI filtering, EVPN L3, EVPN-IRB.

9.7.1. EVPN Bridging

Node 1 is configured with an EVPN bridging service (E-LAN service):

- Local attachment circuit: Ethernet interface from CE-A

- Local End.DT2U bound to a local layer2 table T1 where EVPN is enable: A1::D2AA. That SID is used to attract unicast traffic

- Local End.DT2M bound to the same local layer2 table T1 where EVPN is enable: A1::D2AF:0. That SID is used to attract BUM traffic

Node 4 is configured with an EVPN bridging service:

- Local attachment circuit: Ethernet interface from CE-B

- Local End.DT2U bound to a local layer2 table T1 where EVPN is enable: A4::D2BA. That SID is used to attract unicast traffic

- Local End.DT2M bound to the same local layer2 Table T1 where EVPN is enable: A4::D2BF:0. That SID is used to attract BUM traffic

Node 8 is configured with an EVPN bridging service:

- Local attachment circuit: Ethernet interface from CE-C

- Local End.DT2U bound to a local layer2 table T1 where EVPN is enable: A8::D2CA. That SID is used to attract unicast traffic

- Local End.DT2M bound to the same local layer2 Table T1 where EVPN is enable: A8::D2CF:0/112. That SID is used to attract BUM traffic

The End.DT2M SID are exchanged between nodes via BGP-based EVPN Type-3 route.

Upon reception of EVPN Type-3 routes, each node build its own replication list per layer2 table T1.

On node 1, the replication list looks like: A4::D2BF:0, A8::D2CF:0. On node 4, the replication list looks like: A1::D2AF:0, A8::D2CF:0. On node 8, the replication list looks like: A1:D2AF:0, A4:D2BF:0. In the case of ingress replication, Ingress PE transmitting the BUM traffic stream replicates the traffic using that list.

When node 1 receives a BUM frame F from CE-A, it replicates that frame to remote nodes. For node 4, it pushes an outer IPv6 header with SA=A1::0, DA=A4::D2BF:0 and NH=59. For node 8, it performs the same operation but DA=A8::D2CF:0. Note that no additional header is pushed. Node 1 then forwards the resulting packet on the shortest path for each replication e.g. A4::D2BF:0/112 and A8::D2CF:0/112.

When node 4 receives the packet, it matches the DA in its My LocalSID table and finds the bound function End.DT2M and its related layer2 table T1. After confirming that the next-header=59, node 4 decaps

the outer IPv6 header and forwards the inner Ethernet frame to all layer-2 output interface found to table T1. Similar processing is also performed by node 8 upon packet reception. This example is the same for any BUM stream coming from CE-B and CE-C.

Node 1,4 and 8 are also performing software MAC learning to exchange MAC reachability information (unicast traffic) via BGP among themselves.

Each MAC being learned in software are exchanged using BGP-based EVPN route type-2.

When node 1 receives an unicast frame F from CE-A, it learns its MAC-SA=CEA in software. Node 1 transmits that MAC and its associated SID A1::D2AA using BGP-based EVPN route-type 2 to all remote nodes.

When node 4 receives an unicast frame F from CE-B destinated to MAC-DA=CEA, it performs a L2 table T1 MAC-DA lookup to find the associated SID. It pushes an outer IPv6 header with SA=A4::0, DA=A1::D2AA and NH=59. Note that no additional header is pushed. Node 4 then forwards the resulting packet on the shortest path to A1::/40. Similar processing is also performed by node 8.

9.7.2. EVPN Multi-homing with ESI filtering

In L2 network, traffic loop avoidance is a MUST. In EVPN all-active multi-homing scenario, ESI filtering feature enforce that requirement.

Node 1 and node 2 are peering partners of a redundancy group where the access CE-A is connected in an all-active multi-homing way with these two nodes.

Node 1 is configured with an EVPN bridging service (E-LAN service):

- Local attachment circuit: Ethernet interface from CE-A

- Local Arg.FE2 bound to the attachment circuit: 0xC1

- Local End.DT2M bound to the same local layer2 table T1 where EVPN is enable: A1::D2AF:0/112. That SID is used to attract BUM traffic

Node 2 is configured with an EVPN bridging service:

- Local attachment circuit: Ethernet interface from CE-A

- Local Arg.FE2 bound to the attachment circuit: 0xC2

- Local End.DT2M bound to the same local layer2 Table T1 where EVPN is enable: A2::D2BF:0/112. That SID is used to attract BUM traffic

The End.DT2M SID are exchanged between nodes via BGP-based EVPN route type-3.

Upon reception of EVPN Type-3 routes, each node build its own replication list per layer2 table T1.

The End.DT2M SID arguments Arg.FE2 are exchange between nodes via BGP ESI-filtering extended community attached to BGP-based EVPN route type-1.

Upon reception of EVPN route type-1 and type-3, node 1 merges the End.DT2M SID and the Arg.FE2 argument from node 2; its peering partner. Its replication list looks like A2::D2BF:C1. Similar procedure is performed by node 2.

When node 1 receives a BUM frame F from CE-A, it replicates that frame to remote nodes. For node 2, it pushes an outer IPv6 header with SA=A1::0, DA=A2::D2BF:C1 and NH=59. Note that no additional header is pushed. Node 1 then forwards the resulting packet on the shortest path for each replication e.g. A2::D2BF:00/112. Again, similar processing is also performed by node 8 upon packet reception

9.7.3. EVPN Layer-3

EVPN layer-3 works exactly in the same way of IPVPN. Please refer to SR-IPVPN section

9.7.4. EVPN Integrated Routing Bridging (IRB)

EVPN IRB brings Layer-2 and Layer-3 together. It uses BGP-based EVPN route type-2 to achieve Layer-2 intra-subnet and Layer-3 inter-subnet forwarding. The EVPN route type-2 maintain the associated of a MAC/ IP association.

Node 1 is configured with an EVPN IRB service:

- Local attachment circuit: Ethernet interface from CE-A

- Local End.DT2U bound to a local layer2 table T1 where EVPN is enable: SID = A1::D2AA. That SID is used to attract unicast L2 traffic

- Local End.DT2 bound to tenant IPv4 table 100: SID = A1::D3AA. That SID is used to attract L3 traffic

Node 8 is configured with an EVPN IRB service:

- Local attachment circuit: Ethernet interface from CE-C

- Local End.DT2U bound to a local layer2 table T1 where EVPN is enable: SID = A8::D2CB. That SID is used to attract unicast L2 traffic

- Local End.DT2 bound to tenant IPv4 table 100: SID = A8::D3CB. That SID is used to attract L3 traffic

Each ARP/ND request learned by each node are exchanged using BGP-based EVPN route type-2.

When node 1 receives an ARP/ND packet P from a host (10.10.10.10) on CE-A destined to 20.20.20.20, it learns its MAC-SA=CEA in software. It also learns the ARP/ND entry (IP SA=10.10.10.10) in its cache. Node 1 transmits that MAC/IP and its associated L2 SID A1::D2AA and L3 SID A1::D3AA using BGP-based EVPN route-type 2 to all remote nodes.

When node 8 receives a packet P from CE-C destined to 10.10.10.10 from a host (20.20.20.20), P looks up its tenant-100 IPv4 table and finds an SR-VPN entry for that prefix. As a consequence, node 8 pushes an outer IPv6 header with SA=A8::0, DA=A1::D3AA and NH=4. Node 8 then forwards the resulting packet on the shortest path to A1::/40. EVPN inter-subnet forwarding is then achieved.

When node 8 receives a packet P from CE-C destined to 10.10.10.10 from a host (10.10.10.11), P looks up its L2 table T1 MAC-DA lookup to find the associated SID. It pushes an outer IPv6 header with SA=A8::0, DA=A1::D2AA and NH=59. Note that no additional header is pushed. Node 8 then forwards the resulting packet on the shortest path to A1::/40. EVPN intra-subnet forwarding is then achieved.

9.8. SR TE for Underlay SLA

9.8.1. SR policy from the Ingress PE

Let’s assume that node 1’s tenant-100 IPv4 route "20/8 via A8::D100" is programmed with a color/community that requires low-latency underlay optimization [I-D.filsfils-spring-segment-routing-policy].

In such case, node 1 either computes the low-latency path to the egress node itself or delegates the computation to a PCE.

In either case, the location of the egress PE can easily be found by looking for who originates the SID block comprising the SID A8::D100.

This can be found in the IGP's LSDB for a single domain case, and in the BGP-LS LSDB for a multi-domain case.

Let us assume that the TE metric encodes the per-link propagation latency. Let us assume that all the links have a TE metric of 10, except link 27 which has TE metric 100.

The low-latency path from 1 to 8 is thus 1245678.

This path is encoded in a SID list as: first a hop through A4::C5 and then a hop to 8.

As a consequence the SR-VPN entry 20/8 installed in the Node1's Tenant-100 IPv4 table is: T.Encaps with SRv6 Policy <A4::C5, A8::D100>.

When 1 receives a packet P from CE-A destined to 20.20.20.20, P looks up its tenant-100 IPv4 table and finds an SR-VPN entry 20/8. As a consequence, 1 pushes an outer header with SA=A1::0, DA=A4::C5, NH=SRH followed by SRH (A8::D100, A4::C5; SL=1; NH=4). 1 then forwards the resulting packet on the interface to 2.

2 forwards to 4 along the path to A4::/40.

When 4 receives the packet, 4 matches the DA in its My LocalSID table and finds the bound function End.X to neighbor 5. 4 notes the PSP capability of the SID A4::C5. 4 sets the DA to the next SID A8::D100. As 4 is the penultimate segment hop, it performs PSP and pops the SRH. 4 forwards the resulting packet to 5.

5, 6 and 7 forwards along the path to A8::/40.

When 8 receives the packet, 8 matches the DA in its My LocalSID Table and finds the bound function End.DT(100). As a result, 8 decaps the outer header, looks up the inner IPv4 DA in tenant-100 IPv4 table, and forward the (inner) IPv4 packet towards CE-B.

9.8.2. SR policy at a midpoint

Let us analyze a policy applied at a midpoint on a packet without SRH.

Packet P1 is (A1::, A8::D100).

Let us consider P1 when it is received by node 2 and let us assume that that node 2 is configured to steer A8::/40 in a transit behavior T.Insert associated with SR policy <A4::C5>.

In such a case, node 2 would send the following modified packet P1 on the link to 4:

(A1::, A4::C5)(A8::D100, A4::C5; SL=1).

The rest of the processing is similar to the previous section.

Let us analyze a policy applied at a midpoint on a packet with an SRH.

Packet P2 is (A1::, A7::1)(A8::D100, A7::1; SL=1).

Let us consider P2 when it is received by node 2 and let us assume that node 2 is configured to steer A7::/40 in a transit behavior T.Insert associated with SR policy <A4::C5, A9::1>.

In such a case, node 2 would send the following modified packet P2 on the link to 4:

(A1::, A4::C5)(A7::1, A9::1, A4::C5; SL=2)(A8::D100, A7::1; SL=1)

Node 4 would send the following packet to 5: (A1::, A9::1)(A7::1, A9::1, A4::C5; SL=1)(A8::D100, A7::; SL=1)

Node 5 would send the following packet to 9: (A1::, A9::1)(A7::1, A9::1, A4::C5; SL=1)(A8::D100, A7::1; SL=1)

Node 9 would send the following packet to 6: (A1::, A7::1)(A8::D100, A7::1; SL=1)

Node 6 would send the following packet to 7: (A1::, A7::1)(A8::D100, A7::1; SL=1)

Node 7 would send the following packet to 8: (A1::, A8::D100)

9.9. End-to-End policy with intermediate BSID

Let us now describe a case where the ingress VPN edge node steers the packet destined to 20.20.20.20 towards the egress edge node connected to the tenant100 site with 20/8, but via an intermediate SR Policy represented by a single routable Binding SID. Let us illustrate this case with an intermediate policy which both encodes underlay optimization for low-latency and the service programming via two SR-aware container-based apps.

Let us assume that the End.B6 SID A2::B1 is configured at node 2 and is associated with midpoint T.Insert policy <A4::C5, A9::A1, A6::A2>.

A4::C5 realizes the low-latency path from the ingress PE to the egress PE. This is the underlay optimization part of the intermediate policy.

A9::A1 and A6::A2 represent two SR-aware NFV applications residing in containers respectively connected to node 9 and 6.

Let us assume the following ingress VPN policy for 20/8 in tenant 100 IPv4 table of node 1: T.Encaps with SRv6 Policy <A2::B1, A8::D100>.

This ingress policy will steer the 20/8 tenant-100 traffic towards the correct egress PE and via the required intermediate policy that realizes the SLA and NFV requirements of this tenant customer.

Node 1 sends the following packet to 2: (A1::, A2::B1) (A8::D100, A2::B1; SL=1)

Node 2 sends the following packet to 4: (A1::, A4::C5) (A6::A2, A9::A1, A4::C5; SL=2)(A8::D100, A2::B1; SL=1)

Node 4 sends the following packet to 5: (A1::, A9::A1) (A6::A2, A9::A1, A4::C5; SL=1)(A8::D100, A2::B1; SL=1)

Node 5 sends the following packet to 9: (A1::, A9::A1) (A6::A2, A9::A1, A4::C5; SL=1)(A8::D100, A2::B1; SL=1)

Node 9 sends the following packet to 6: (A1::, A6::A2) (A8::D100, A2::B1; SL=1)

Node 6 sends the following packet to 7: (A1::, A8::D100)

Node 7 sends the following packet to 8: (A1::, A8::D100) which decaps and forwards to CE-B.

The benefits of using an intermediate Binding SID are well-known and key to the Segment Routing architecture: the ingress edge node needs to push fewer SIDs, the ingress edge node does not need to change its SR policy upon change of the core topology or re-homing of the container-based apps on different servers. Conversely, the core and service organizations do not need to share details on how they realize underlay SLA’s or where they home their NFV apps.

9.10. TI-LFA

Let us assume two packets P1 and P2 received by node 2 exactly when the failure of link 27 is detected.

P1: (A1::, A7::1)

P2: (A1::, A7::1)(A8::D100, A7::1; SL=1)

Node 2's pre-computed TI-LFA backup path for the destination A7:: is <A4::C5>. It is installed as a T.Insert transit behavior.

Node 2 protects the two packets P1 and P2 according to the pre-computed TI-LFA backup path and send the following modified packets on the link to 4:

P1: (A1::, A4::C5)(A7::1, A4::C5; SL=1)

P2: (A1::, A4::C5)(A7::1, A4::C5; SL=1) (A8::D100, A7::1; SL=1)

Node 4 then sends the following modified packets to 5:

P1: (A1::, A7::1)

P2: (A1::, A7::1)(A8::D100, A7::1; SL=1)

Then these packets follow the rest of their post-convergence path towards node 7 and then go to node 8 for the VPN decaps.

9.11. SR TE for Service programming

We have illustrated the service programming through SR-aware apps in a previous section.

We illustrate the use of End.AS function [I-D.xuclad-spring-sr-service-programming] to service chain an IP flow bound to the internet through two SR-unaware applications hosted in containers.

Let us assume that servers 20 and 70 are respectively connected to nodes 2 and 7. They are respectively configured with SID spaces A020::/40 and A070::/40. Their connected routers advertise the related prefixes in the IGP. Two SR-unaware container-based applications App2 and App7 are respectively hosted on server 20 and 70. Server 20 (70) is configured explicitly with an End.AS SID A020::2 for App2 (A070::7 for App7).

Let us assume a broadband customer with a home gateway CE-A connected to edge router 1. Router 1 is configured with an SR policy which encapsulates all the traffic received from CE-A into a T.Encaps policy <A020::2, A070::7, A8::D0> where A8::D0 is an End.DT4 SID instantiated at node 8.

P1 is a packet sent by the broadband customer to 1: (X, Y) where X and Y are two IPv4 addresses.

1 sends the following packet to 2: (A1::0, A020::2)(A8::D0, A070::7, A020::2; SL=2; NH=4)(X, Y).

2 forwards the packet to server 20.

20 receives the packet (A1::0, A020::2)(A8::D0, A070::7, A020::2; SL=2; NH=4)(X, Y) and forwards the inner IPv4 packet (X,Y) to App2. App2 works on the packet and forwards it back to 20. 20 pushes the outer IPv6 header with SRH (A1::0, A070::7)(A8::D0, A070::7, A020::2; SL=1; NH=4) and sends the (whole) IPv6 packet with the encapsulated IPv4 packet back to 2.

2 and 7 forward to server 70.

70 receives the packet (A1::0, A070::7)(A8::D0, A070::7, A020::2; SL=1; NH=4)(X, Y) and forwards the inner IPv4 packet (X,Y) to App7. App7 works on the packet and forwards it back to 70. 70 pushes the outer IPv6 header with SRH (A1::0, A8::D0)(A8::D0, A070::7, A020::2; SL=0; NH=4) and sends the (whole) IPv6 packet with the encapsulated IPv4 packet back to 7.

7 forwards to 8.

8 receives (A1::0, A8::D0)(A8::D0, A070::7, A020::2; SL=0; NH=4)(X, Y) and performs the End.DT4 function and sends the IP packet (X, Y) towards its internet destination.

10. Benefits

10.1. Seamless deployment

The VPN use-case can be realized with SRv6 capability deployed solely at the ingress and egress PE’s.

All the nodes in between these PE’s act as transit routers as per [RFC8200]. No software/hardware upgrade is required on all these nodes. They just need to support IPv6 per [RFC8200].

The SRTE/underlay-SLA use-case can be realized with SRv6 capability deployed at few strategic nodes.

It is well-known from the experience deploying SR-MPLS that underlay SLA optimization requires few SIDs placed at strategic locations. This was illustrated in our example with the low-latency optimization which required the operator to enable one single core node with SRv6 (node 4) where one single and End.X SID towards node 5 was instantiated. This single SID is sufficient to force the end-to-end traffic via the low-latency path.

The TI-LFA benefits are collected incrementally as SRv6 capabilities are deployed.

It is well-know that TI-LFA is an incremental node-by-node deployment. When a node N is enabled for TI-LFA, it computes TI-LFA backup paths for each primary path to each IGP destination. In more than 50% of the case, the post-convergence path is loop-free and does not depend on the presence of any remote SRv6 SID. In the vast majority of cases, a single segment is enough to encode the post-convergence path in a loop-free manner. If the required segment is available (that node has been upgraded) then the related back-up path is installed in FIB, else the pre-existing situation (no backup) continues. Hence, as the SRv6 deployment progresses, the coverage incrementally increases. Eventually, when the core network is SRv6 capable, the TI-LFA coverage is complete.

The service programming use-case can be realized with SRv6 capability deployed at few strategic nodes.

The service-programming deployment is again incremental and does not require any pre-deployment of SRv6 in the network. When an NFV app A1 needs to be enabled for inclusion in an SRv6 service chain, all what is required is to install that app in a container or VM on an SRv6-capable server (Linux 4.10 or FD.io 17.04 release). The app can either be SR-aware or not, leveraging the proxy functions described in this document.

By leveraging the various END functions it can also be used to support any current PNF/VNF implementations and their forwarding methods (e.g. Layer 2).

The ability to leverage SR TE policies and BSIDs also permits building scalable, hierarchical service-chains.

10.2. Integration

The SRv6 network programming concept allows integrating all the application and service requirements: multi-domain underlay SLA optimization with scale, overlay VPN/Tenant, sub-50msec automated FRR, security and service programming.

10.3. Security

The combination of well-known techniques (SEC1, SEC2, SEC4) and carefully chosen architectural rules (SEC3) ensure a secure deployment of SRv6 inside a multi-domain network managed by a single organization.

Inter-domain security will be described in a companion document.

11. IANA Considerations

This document requests the following new IANA registries:

- A new top-level registry "Segment-routing with IPv6 dataplane (SRv6) Parameters" to be created under IANA Protocol registries. This registry is being defined to serve as a top-level registry for keeping all other SRv6 sub-registries.

- A sub-registry "SRv6 Endpoint Types" to be defined under top-level "Segment-routing with IPv6 dataplane (SRv6) Parameters" registry. This sub-registry maintains 16-bit code-points for the defined SRv6 Endpoint types. The range of the registry is 0-65535 (0x0000 - 0xFFFF) and has the following registration rules and allocation policies:

Range	Hex	Registration proceadure	Notes
0	0x0000	Reserved	Invalid
1-32767	0x0001-0x7FFF	IETF review	Draft Specifications
32768-49151	0x8000-0xBFFF	Reserved for experimental use	
49152-65534	0xC000-0xFFFE	Reserved for private use	
65535	0xFFFF	Reserved	Wildcard

Table 3: SRv6 Endpoint Types

The initial registrations for the "Draft Specifications" portion of the sub-registry are as follows:

Value	Hex	Endpoint function	Reference
1	0x0001	End (no PSP, no USP)	[This.ID]
2	0x0002	End with PSP	[This.ID]
3	0x0003	End with USP	[This.ID]
4	0x0004	End with PSP&USP	[This.ID]
5	0x0005	End.X (no PSP, no USP)	[This.ID]
6	0x0006	End.X with PSP	[This.ID]
7	0x0007	End.X with USP	[This.ID]
8	0x0008	End.X with PSP&USP	[This.ID]
9	0x0009	End.T (no PSP, no USP)	[This.ID]
10	0x000A	End.T with PSP	[This.ID]
11	0x000B	End.T with USP	[This.ID]
12	0x000C	End.T with PSP&USP	[This.ID]
13	0x000D	End.B6	[This.ID]
14	0x000E	End.B6.Encaps	[This.ID]
15	0x000F	End.BM	[This.ID]
16	0x0010	End.DX6	[This.ID]
17	0x0011	End.DX4	[This.ID]
18	0x0012	End.DT6	[This.ID]
19	0x0013	End.DT4	[This.ID]
20	0x0014	End.DT46	[This.ID]
21	0x0015	End.DX2	[This.ID]
22	0x0016	End.DX2V	[This.ID]
23	0x0017	End.DT2U	[This.ID]
24	0x0018	End.DT2M	[This.ID]
25	0x0019	End.S	[This.ID]
26	0x001A	End.B6.Red	[This.ID]
27	0x001B	End.B6.Encaps.Red	[This.ID]

Table 4: SRv6 Endpoint Types

12. Work in progress

We are working on a extension of this document to provide Yang modelling for all the functionality described in this document. This work is ongoing in [I-D.raza-spring-srv6-yang].

13. Acknowledgements

The authors would like to acknowledge Stefano Previdi, Dave Barach, Mark Townsley, Peter Psenak, Paul Wells, Robert Hanzl, Dan Ye, Gaurav Dawra, Faisal Iqbal, Jaganbabu Rajamanickam, David Toscano, Asif Islam, Jianda Liu, Yunpeng Zhang, Jiaoming Li, Narendra A.K, Mike Mc Gourty, Bhupendra Yadav, Sherif Toulan, Satish Damodaran, John

Bettink, Kishore Nandyala Veera Venk, Jisu Bhattacharya and Saleem Hafeez.

14. Contributors

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Dirk Steinberg
Steinberg Consulting
Germany

Email: dws@dirksteinberg.de

Robert Raszuk
Bloomberg LP
United States of America

Email: robert@raszuk.net

Bruno Decraene
Orange
Frence

Email: bruno.decraene@orange.com

Bart Peirens
Proximus
Belgium

Email: bart.peirens@proximus.com

Hani Elmalky
Ericsson
United States of America

Email: hani.elmalky@gmail.com

Prem Jonnalagadda
Barefoot Networks
United States of America

Email: prem@barefootnetworks.com

Milad Sharif

Barefoot Networks
United States of America

Email: msharif@barefootnetworks.com

David Lebrun
Universite catholique de Louvain
Belgium

Email: david.lebrun@uclouvain.be

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

Ahmed AbdelSalam
Gran Sasso Science Institute
Italy

Email: ahmed.abdelsalam@gssi.it

Gaurav Naik
Drexel University
United States of America

Email: gn@drexel.edu

Arthi Ayyangar
Arista
United States of America

Email: arthi@arista.com

Satish Mynam
Innovium Inc.
United States of America

Email: smynam@innovium.com

Wim Henderickx
Nokia
Belgium

Email: wim.henderickx@nokia.com

Shaowen Ma

Juniper
Singapore

Email: mashao@juniper.net

Ahmed Bashandy
Individual
United States of America

Email: abashandy.ietf@gmail.com

Francois Clad
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Kamran Raza
Cisco Systems, Inc.
Canada

Email: skraza@cisco.com

Darren Dukes
Cisco Systems, Inc.
Canada

Email: ddukes@cisco.com

Patrice Brissete
Cisco Systems, Inc.
Canada

Email: pbrisset@cisco.com

Zafar Ali
Cisco Systems, Inc.
United States of America

Email: zali@cisco.com

15. References

15.1. Normative References

[I-D.ali-spring-srv6-oam]
Ali, Z., Filsfils, C., Kumar, N., Pignataro, C., faiqbal@cisco.com, f., Gandhi, R., Leddy, J., Matsushima, S., Raszuk, R., Peirens, B., and G. Naik, "Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)", draft-ali-spring-srv6-oam-00 (work in progress), February 2018.

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-14 (work in progress), June 2018.

[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>.

15.2. Informative References

[I-D.bashandy-isis-srv6-extensions]
Ginsberg, L., Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Routing over IPv6 Dataplane", draft-bashandy-isis-srv6-extensions-03 (work in progress), June 2018.

[I-D.dawra-idr-srv6-vpn]
Dawra, G., Filsfils, C., Dukes, D., Brissette, P., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Decraene, B., Matsushima, S., and S. Zhuang, "BGP Signaling of IPv6-Segment-Routing-based VPN Networks", draft-dawra-idr-srv6-vpn-04 (work in progress), June 2018.

[I-D.filsfils-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Hegde, S., daniel.voyer@bell.ca, d., Lin, S., bogdanov@google.com, b., Krol, P., Horneffer, M., Steinberg, D., Decraene, B., Litkowski, S., Mattes, P., Ali, Z., Talaulikar, K., Liste, J., Clad, F., and K. Raza, "Segment Routing Policy Architecture", draft-filsfils-spring-segment-routing-policy-06 (work in progress), May 2018.

[I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H., and M. Chen, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-08 (work in progress), May 2018.

[I-D.ietf-idr-te-lsp-distribution]
Previdi, S., Talaulikar, K., Dong, J., Chen, M., Gredler, H., and J. Tantsura, "Distribution of Traffic Engineering (TE) Policies and State using BGP-LS", draft-ietf-idr-te-lsp-distribution-09 (work in progress), June 2018.

[I-D.ietf-isis-l2bundles]
Ginsberg, L., Bashandy, A., Filsfils, C., Nanduri, M., and E. Aries, "Advertising L2 Bundle Member Link Attributes in IS-IS", draft-ietf-isis-l2bundles-07 (work in progress), May 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[I-D.raza-spring-srv6-yang]
Raza, K., Rajamanickam, J., Liu, X., Hu, Z., Hussain, I., Shah, H., daniel.voyer@bell.ca, d., Elmalky, H., Matsushima, S., Horiba, K., and A. Abdelsalam, "YANG Data Model for SRv6 Base and Static", draft-raza-spring-srv6-yang-01 (work in progress), March 2018.

[I-D.xuclad-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., Bernier, D., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", draft-xuclad-spring-sr-service-programming-00 (work in progress), July 2018.

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

[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>.

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

[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>.

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Pablo Camarillo Garvia (editor)
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

John Leddy
Comcast
United States of America

Email: john_leddy@cable.comcast.com

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Satoru Matsushima
SoftBank
1-9-1,Higashi-Shimbashi,Minato-Ku
Tokyo 105-7322
Japan

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

Zhenbin Li
Huawei Technologies
China

Email: lizhenbin@huawei.com

SPRING
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2019

C. Filsfils
P. Camarillo, Ed.
Cisco Systems, Inc.
J. Leddy
Comcast
D. Voyer
Bell Canada
S. Matsushima
SoftBank
Z. Li
Huawei Technologies
February 14, 2019

SRv6 Network Programming
draft-filsfils-spring-srv6-network-programming-07

Abstract

This document describes the SRv6 network programming concept and its most basic functions.

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.

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 August 18, 2019.

Table of Contents

1. Introduction

Segment Routing leverages the source routing paradigm. An ingress node steers a packet through a ordered list of instructions, called segments. Each one of these instructions represents a function to be called at a specific location in the network. A function is locally defined on the node where it is executed and may range from simply moving forward in the segment list to any complex user-defined behavior. The network programming consists in combining segment routing functions, both simple and complex, to achieve a networking objective that goes beyond mere packet routing.

This document defines the SRv6 Network Programming concept and aims at standardizing the main segment routing functions to enable the creation of interoperable overlays with underlay optimization and service programming.

The companion document [I-D.filsfils-spring-srv6-net-pgm-illustration] illustrates the concepts defined in this document.

Familiarity with the Segment Routing Header [I-D.ietf-6man-segment-routing-header] is assumed.

2. Terminology

SRH is the abbreviation for the Segment Routing Header. We assume that the SRH may be present multiple times inside each packet.

NH is the abbreviation of the IPv6 next-header field.

NH=SRH means that the next-header field is 43 with routing type 4.

When there are multiple SRHs, they must follow each other: the next-header field of all SRH, except the last one, must be SRH.

The effective next-header (ENH) is the next-header field of the IP header when no SRH is present, or is the next-header field of the last SRH.

In this version of the document, we assume that there are no other extension headers than the SRH. These will be lifted in future versions of the document.

SID: A Segment Identifier which represents a specific segment in segment routing domain. The SID type used in this document is IPv6 address (also referenced as SRv6 Segment or SRv6 SID).

A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SR path.

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

- IPv6 header with source address SA, destination addresses DA and SRH as next-header

- SRH with SID list <S1, S2, S3> with SegmentsLeft = SL

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

- The payload of the packet is omitted.

SRH[SL] represents the SID pointed by the SL field in the first SRH. In our example, SRH[2] represents S1, SRH[1] represents S2 and SRH[0] represents S3.

FIB is the abbreviation for the forwarding table. A FIB lookup is a lookup in the forwarding table.

When a packet is intercepted on a wire, it is possible that SRH[SL] is different from the DA.

3. SRv6 Segment

An SRv6 Segment is a 128-bit value. "SID" (abbreviation for Segment Identifier) is often used as a shorter reference for "SRv6 Segment".

An SRv6-capable node N maintains a "My SID Table". This table contains all the SRv6 segments explicitly instantiated at node N. N is the parent node for these SIDs.

A local SID of N can be an IPv6 address associated to a local interface of N but it is not mandatory. Nor is the "My SID table" populated by default with all IPv6 addresses defined on node N.

In most use-cases, a local SID will NOT be an address associated to a local interface of N.

A local SID of N could be routed to N but it does not have to be. Most often, it is routed to N via a shorter-mask prefix.

Let’s provide a classic illustration.

Node N is configured with a loopback0 interface address of A:1::/32 originated in its IGP. Node N is configured with two SIDs: B:1:100:: and B:2:101::.

The entry A:1:: is not defined explicitly as an SRv6 SID and hence does not appear in the "My SID Table". The entries B:1:100:: and B:2:101:: are defined explicitly as SRv6 SIDs and hence appear in the "My SID Table".

The network learns about a path to B:1::/32 via the IGP and hence a packet destined to B:1:100:: would be routed up to N. The network does not learn about a path to B:2::/32 via the IGP and hence a packet destined to B:2:101:: would not be routed up to N.

A packet could be steered to a non-routed SID B:2:101:: by using a SID list <...,B:1:100::,B:2:101::,...> where the non-routed SID is preceded by a routed SID to the same node. This is similar to the local vs global segments in SR-MPLS.

Every SRv6 SID instantiated has a specific instruction bound to it. This information is stored in the "My SID Table". The "My SID Table" has three main purposes:

- Define which SIDs are explicitly instantiated on that node

- Specify which instruction is bound to each of the instantiated SIDs

- Store the parameters associated with such instruction (i.e. OIF, NextHop, VRF,...)

We represent an SRv6 SID as LOC:FUNCT where LOC is the L most significant bits and FUNCT is the 128-L least significant bits. L is called the locator length and is flexible. Each operator is free to use the locator length it chooses. Most often the LOC part of the SID is routable and leads to the node which instantiates that SID.

The FUNCT part of the SID is an opaque identification of a local function bound to the SID. The FUNCT value zero is invalid.

Often, for simplicity of illustration, we will use a locator length of 32 bits. This is just an example. Implementations must not assume any a priori prefix length.

A function may require additional arguments that would be placed immediately after the FUNCT. In such case, the SRv6 SID will have the form LOC:FUNCT:ARGS::. For this reason, the "My SID Table" matches on a per longest-prefix-match basis.

These arguments may vary on a per-packet basis and may contain information related to the flow, service, or any other information required by the function associated to the SRv6 SID.

A node may receive a packet with an SRv6 SID in the DA without an SRH. In such case the packet should still be processed by the Segment Routing engine.

4. Functions associated with a SID

Each entry of the "My SID Table" indicates the function associated with the local SID and its parameters.

We define hereafter a set of well-known functions that can be associated with a SID.

End	Endpoint function The SRv6 instantiation of a prefix SID
End.X	Endpoint with Layer-3 cross-connect The SRv6 instantiation of a Adj SID
End.T	Endpoint with specific IPv6 table lookup
End.DX2	Endpoint with decaps and L2 cross-connect e.g. L2VPN use-case
End.DX2V	Endpoint with decaps and VLAN L2 table lookup EVPN Flexible cross-connect use-cases
End.DT2U	Endpoint with decaps and unicast MAC L2table lookup EVPN Bridging unicast use-cases
End.DT2M	Endpoint with decaps and L2 table flooding EVPN Bridging BUM use-cases with ESI filtering
End.DX6	Endpoint with decaps and IPv6 cross-connect e.g. IPv6-L3VPN (equivalent to per-CE VPN label)
End.DX4	Endpoint with decaps and IPv4 cross-connect e.g. IPv4-L3VPN (equivalent to per-CE VPN label)
End.DT6	Endpoint with decaps and IPv6 table lookup e.g. IPv6-L3VPN (equivalent to per-VRF VPN label)
End.DT4	Endpoint with decaps and IPv4 table lookup e.g. IPv4-L3VPN (equivalent to per-VRF VPN label)
End.DT46	Endpoint with decaps and IP table lookup e.g. IP-L3VPN (equivalent to per-VRF VPN label)
End.B6.Insert	Endpoint bound to an SRv6 policy SRv6 instantiation of a Binding SID
End.B6.Insert.RED	[...] with reduced SRH insertion SRv6 instantiation of a Binding SID
End.B6.Encaps	Endpoint bound to an SRv6 policy with encaps SRv6 instantiation of a Binding SID
End.B6.Encaps.RED	[...] with reduced SRH insertion SRv6 instantiation of a Binding SID
End.BM	Endpoint bound to an SR-MPLS Policy SRv6 instantiation of an SR-MPLS Binding SID
End.S	Endpoint in search of a target in table T

The list is not exhaustive. In practice, any function can be attached to a local SID: e.g. a node N can bind a SID to a local VM or container which can apply any complex function on the packet.

We call N the node who has an explicitly instantiated SID S and we detail the function that N binds to S.

At the end of this section we also present some flavours of these well-known functions.

4.1. End: Endpoint

The Endpoint function ("End" for short) is the most basic function.

When N receives a packet whose IPv6 DA is S and S is a local End SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      decrement SL
3.      update the IPv6 DA with SRH[SL]
4.      FIB lookup on the updated DA                              ;; Ref1
5.      forward accordingly to the matched entry                  ;; Ref2
6.   ELSE IF NH!=SRH
7.      Send an ICMP parameter problem message; drop the packet   ;; Ref3
8.   ELSE
9.      drop the packet
```

Ref1: The End function performs the FIB lookup in the forwarding table associated to the ingress interface

Ref2: If the FIB lookup matches a multicast state, then the related RPF check must be considered successful

Ref3: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

A local SID could be bound to a function which authorizes the decapsulation of an outer header (e.g. IPinIP) or the punting of the packet to TCP, UDP or any other protocol. This however needs to be explicitly defined in the function bound to the local SID. By default, a local SID bound to the well-known function "End"neither allows the decapsulation of an outer header nor the cleanup of an SRH. As a consequence, an End SID cannot be the last SID of an SRH and cannot be the DA of a packet without SRH.

This is the SRv6 instantiation of a Prefix SID [I-D.ietf-spring-segment-routing].

4.2. End.X: Layer-3 cross-connect

The "Endpoint with cross-connect to an array of layer-3 adjacencies" function (End.X for short) is a variant of the End function.

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

```
1.   IF NH=SRH and SL > 0
2.      decrement SL
3.      update the IPv6 DA with SRH[SL]
4.      forward to layer-3 adjacency bound to the SID S         ;; Ref1
6.   ELSE IF NH!=SRH
7.      Send an ICMP parameter problem message; drop the packet ;; Ref2
8.   ELSE
9.      drop the packet
```

Ref1: If an array of adjacencies is bound to the End.X SID, then one entry of the array is selected based on a hash of the packet's header.

Ref2: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

The End.X function is required to express any traffic-engineering policy.

This is the SRv6 instantiation of an Adjacency SID [I-D.ietf-spring-segment-routing].

If a node N has 30 outgoing interfaces to 30 neighbors, usually the operator would explicitly instantiate 30 End.X SIDs at N: one per layer-3 adjacency to a neighbor. Potentially, more End.X could be explicitly defined (groups of layer-3 adjacencies to the same neighbor or to different neighbors).

Note that with SR-MPLS, an AdjSID is typically preceded by a PrefixSID. This is unlikely in SRv6 as most likely an End.X SID is globally routed to N.

Note that if N has an outgoing interface bundle I to a neighbor Q made of 10 member links, N may allocate up to 11 End.X local SIDs for that bundle: one for the bundle itself and then up to one for each member link. This is the equivalent of the L2-Link Adj SID in SR-MPLS [I-D.ietf-isis-l2bundles].

4.3. End.T: Specific IPv6 table lookup

The "Endpoint with specific IPv6 table lookup" function (End.T for short) is a variant of the End function.

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

```
1.   IF NH=SRH and SL > 0                                          ;; Ref1
2.      decrement SL
3.      update the IPv6 DA with SRH[SL]
4.      lookup the next segment in IPv6 table T associated with the SID
5.      forward via the matched table entry
6.   ELSE IF NH!=SRH
7.      Send an ICMP parameter problem message; drop the packet ;; Ref2
8.   ELSE
9.      drop the packet
```

Ref1: The End.T SID must not be the last SID

Ref2: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

The End.T is used for multi-table operation in the core.

4.4. End.DX2: Decapsulation and L2 cross-connect

The "Endpoint with decapsulation and Layer-2 cross-connect to OIF" function (End.DX2 for short) is a variant of the endpoint function.

When N receives a packet destined to S and S is a local End.DX2 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                            ;; Ref1
3.   ELSE IF ENH=59                                                ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      forward the resulting frame to OIF bound to the SID S
6.   ELSE
7.      Send an ICMP parameter problem message                     ;; Ref3
8.      drop the packet
```

Ref1: An End.DX2 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: We conveniently reuse the next-header value 59 allocated to IPv6 No Next Header [RFC8200]. When the SID corresponds to function

End.DX2 and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

Ref3: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

An End.DX2 function could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

One of the applications of the End.DX2 function is the L2VPN/EVPN VPWS use-case.

4.5. End.DX2V: Decapsulation and VLAN L2 table lookup

The "Endpoint with decapsulation and specific VLAN table lookup" function (End.DX2V for short) is a variant of the endpoint function.

When N receives a packet destined to S and S is a local End.DX2V SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH  = 59                                     ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      lookup the exposed inner VLANs in L2 table T
6.      forward via the matched table entry
7.   ELSE
8.      Send an ICMP parameter problem message             ;; Ref3
9.      drop the packet
```

Ref1: An End.DX2V SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: We conveniently reuse the next-header value 59 allocated to IPv6 No Next Header [RFC8200]. When the SID corresponds to function End.DX2V and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

Ref3: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

An End.DX2V function could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

The End.DX2V is used for EVPN Flexible cross-connect use-cases.

4.6. End.DT2U: Decapsulation and unicast MAC L2 table lookup

The "Endpoint with decapsulation and specific unicast MAC L2 table lookup" function (End.DT2U for short) is a variant of the endpoint function.

When N receives a packet destined to S and S is a local End.DT2U SID, N does:

```
 1.   IF NH=SRH and SL > 0
 2.      drop the packet                                    ;; Ref1
 3.   ELSE IF ENH  = 59                                     ;; Ref2
 4.      pop the (outer) IPv6 header and its extension headers
 5.      learn the exposed inner MAC SA in L2 table T       ;; Ref3
 6.      lookup the exposed inner MAC DA in L2 table T
 7.      IF matched entry in table T
 8.         forward via the matched table T entry
 9.      ELSE
10.         forward via all L2OIF entries in table T
11.   ELSE
12.      Send an ICMP parameter problem message             ;; Ref4
13.      drop the packet
```

Ref1: An End.DT2U SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: We conveniently reuse the next-header value 59 allocated to IPv6 No Next Header [RFC8200]. When the SID corresponds to function End.DT2U and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

Ref3: In EVPN, the learning of the exposed inner MAC SA is done via control plane.

Ref4: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

The End.DT2U is used for EVPN Bridging unicast use cases.

4.7. End.DT2M: Decapsulation and L2 table flooding

The "Endpoint with decapsulation and specific L2 table flooding" function (End.DT2M for short) is a variant of the endpoint function.

This function may take an argument: "Arg.FE2". It is an argument specific to EVPN ESI filtering. It is used to exclude a specific OIF (or set of OIFs) from L2 table T flooding.

When N receives a packet destined to S and S is a local End.DT2M SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                        ;; Ref1
3.   ELSE IF ENH  = 59                                        ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
3.      learn the exposed inner MAC SA in L2 table T           ;; Ref3
4.      forward on all L2OIF excluding the one specified in Arg.FE2
5.   ELSE
6.      Send an ICMP parameter problem message                 ;; Ref4
7.      drop the packet
```

Ref1: An End.DT2M SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: We conveniently reuse the next-header value 59 allocated to IPv6 No Next Header [RFC8200]. When the SID corresponds to function End.DT2M and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

Ref3: In EVPN, the learning of the exposed inner MAC SA is done via control plane

Ref4: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

The End.DT2M is used for EVPN Bridging BUM use-case with ESI filtering capability.

4.8. End.DX6: Decapsulation and IPv6 cross-connect

The "Endpoint with decapsulation and cross-connect to an array of IPv6 adjacencies" function (End.DX6 for short) is a variant of the End.X function.

When N receives a packet destined to S and S is a local End.DX6 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                        ;; Ref1
3.   ELSE IF ENH = 41                                         ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      forward to layer-3 adjacency bound to the SID S        ;; Ref3
6.   ELSE
7.      Send an ICMP parameter problem message                 ;; Ref4
8.      drop the packet
```

Ref1: The End.DX6 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers

Ref3: Selected based on a hash of the packet's header (at least SA, DA, Flow Label)

Ref4: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

One of the applications of the End.DX6 function is the L3VPNv6 use-case where a FIB lookup in a specific tenant table at the egress PE is not required. This would be equivalent to the per-CE VPN label in MPLS [RFC4364].

4.9. End.DX4: Decapsulation and IPv4 cross-connect

The "Endpoint with decapsulation and cross-connect to an array of IPv4 adjacencies" function (End.DX4 for short) is a variant of the End.X functions.

When N receives a packet destined to S and S is a local End.DX4 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH  =  4                                     ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      forward to layer-3 adjacency bound to the SID S    ;; Ref3
6.   ELSE
7.      Send an ICMP parameter problem message             ;; Ref4
8.      drop the packet
```

Ref1: The End.DX4 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers

Ref3: Selected based on a hash of the packet's header (at least SA, DA, Flow Label)

Ref4: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

One of the applications of the End.DX4 function is the L3VPNv4 use-case where a FIB lookup in a specific tenant table at the egress PE is not required. This would be equivalent to the per-CE VPN label in MPLS [RFC4364].

4.10. End.DT6: Decapsulation and specific IPv6 table lookup

The "Endpoint with decapsulation and specific IPv6 table lookup" function (End.DT6 for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.DT6 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH = 41                                      ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      lookup the exposed inner IPv6 DA in IPv6 table T
6.      forward via the matched table entry
7.   ELSE
8.      Send an ICMP parameter problem message             ;; Ref3
9.      drop the packet
```

Ref1: the End.DT6 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers

Ref3: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

One of the applications of the End.DT6 function is the L3VPNv6 use-case where a FIB lookup in a specific tenant table at the egress PE is required. This would be equivalent to the per-VRF VPN label in MPLS[RFC4364].

Note that an End.DT6 may be defined for the main IPv6 table in which case and End.DT6 supports the equivalent of an IPv6inIPv6 decaps (without VPN/tenant implication).

4.11. End.DT4: Decapsulation and specific IPv4 table lookup

The "Endpoint with decapsulation and specific IPv4 table lookup" function (End.DT4 for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.DT4 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                        ;; Ref1
3.   ELSE IF ENH = 4                                          ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      lookup the exposed inner IPv4 DA in IPv4 table T
6.      forward via the matched table entry
7.   ELSE
8.      Send an ICMP parameter problem message                 ;; Ref3
9.      drop the packet
```

Ref1: the End.DT4 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers

Ref3: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

One of the applications of the End.DT4 is the L3VPNv4 use-case where a FIB lookup in a specific tenant table at the egress PE is required. This would be equivalent to the per-VRF VPN label in MPLS[RFC4364].

Note that an End.DT4 may be defined for the main IPv4 table in which case and End.DT4 supports the equivalent of an IPv4inIPv6 decaps (without VPN/tenant implication).

4.12. End.DT46: Decapsulation and specific IP table lookup

The "Endpoint with decapsulation and specific IP table lookup" function (End.DT46 for short) is a variant of the End.DT4 and End.DT6 functions.

When N receives a packet destined to S and S is a local End.DT46 SID, N does:

```
 1.   IF NH=SRH and SL > 0
 2.      drop the packet                                        ;; Ref1
 3.   ELSE IF ENH  =  4                                         ;; Ref2
 4.      pop the (outer) IPv6 header and its extension headers
 5.      lookup the exposed inner IPv4 DA in IPv4 table T
 6.      forward via the matched table entry
 7.   ELSE IF ENH = 41                                          ;; Ref2
 8.      pop the (outer) IPv6 header and its extension headers
 9.      lookup the exposed inner IPv6 DA in IPv6 table T
10.      forward via the matched table entry
11.   ELSE
12.      Send an ICMP parameter problem message                 ;; Ref3
13.      drop the packet
```

Ref1: the End.DT46 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 4 and 41 refer to IPv4 and IPv6 encapsulation respectively as defined by IANA allocation for Internet Protocol Numbers

Ref3: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

One of the applications of the End.DT46 is the L3VPN use-case where a FIB lookup in a specific IP tenant table at the egress PE is required. This would be equivalent to the per-VRF VPN label in MPLS [RFC4364].

Note that an End.DT46 may be defined for the main IP table in which case and End.DT46 supports the equivalent of an IPinIPv6 decaps (without VPN/tenant implication).

4.13. End.B6.Insert: Endpoint bound to an SRv6 policy

The "Endpoint bound to an SRv6 Policy" is a variant of the End function.

When N receives a packet destined to S and S is a local End.B6.Insert SID, N does:

```
1.   IF NH=SRH and SL > 0                                        ;; Ref1
2.      do not decrement SL nor update the IPv6 DA with SRH[SL]
3.      insert a new SRH, in between the IPv6 header and the
           received SRH
4.      set the IPv6 DA to the first segment of the SRv6 Policy
5.      forward according to the first segment of the SRv6 Policy
6.   ELSE
7.      Send an ICMP parameter problem message                  ;; Ref2
8.      drop the packet
```

Ref1: An End.B6.Insert SID, by definition, is never the last SID.

Ref2: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

Note: Instead of the term "insert", "push" may also be used.

The End.B6.Insert function is required to express scalable traffic-engineering policies across multiple domains. This is the SRv6 instantiation of a Binding SID [I-D.ietf-spring-segment-routing].

4.14. End.B6.Insert.Red: [...] with reduced SRH insertion

This is an optimization of the End.B6.Insert function.

End.B6.Insert.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the pushed SRH. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

4.15. End.B6.Encaps: Endpoint bound to an SRv6 policy w/ encaps

This is a variation of the End.B6.Insert behavior where the SRv6 Policy also includes an IPv6 Source Address A.

When N receives a packet destined to S and S is a local End.B6.Encaps SID, N does:

```
 1.  IF NH=SRH and SL > 0
 2.    decrement SL and update the IPv6 DA with SRH[SL]
 3.    push an outer IPv6 header with its own SRH
 4.    set the outer IPv6 SA to A
 5.    set the outer IPv6 DA to the first segment of the SRv6 Policy
 6.    set outer payload length, trafic class and flow label   ;; Ref1,2
 7.    update the Next-Header value                            ;; Ref1
 8.    decrement inner Hop Limit or TTL                        ;; Ref1
 9.    forward according to the first segment of the SRv6 Policy
10.  ELSE
11.    Send an ICMP parameter problem message                  ;; Ref3
12.    drop the packet
```

Ref 1: As described in [RFC2473] (Generic Packet Tunneling in IPv6 Specification)

Ref 2: As described in [RFC6437] (IPv6 Flow Label Specification)

Ref3: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

Instead of simply inserting an SRH with the policy (End.B6), this behavior also adds an outer IPv6 header. The source address defined for the outer header does not have to be a local SID of the node.

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

4.16. End.B6.Encaps.Red: [...] with reduced SRH insertion

This is an optimization of the End.B6.Encaps function.

End.B6.Encaps.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the outer SRH. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

4.17. End.BM: Endpoint bound to an SR-MPLS policy

The "Endpoint bound to an SR-MPLS Policy" is a variant of the End.B6 function.

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

```
1.   IF NH=SRH and SL > 0                                        ;; Ref1
2.      decrement SL and update the IPv6 DA with SRH[SL]
3.      push an MPLS label stack <L1, L2, L3> on the received packet
4.      forward according to L1
5.   ELSE
6.      Send an ICMP parameter problem message                   ;; Ref2
7.      drop the packet
```

Ref1: an End.BM SID, by definition, is never the last SID.

Ref2: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

The End.BM function is required to express scalable traffic-engineering policies across multiple domains where some domains support the MPLS instantiation of Segment Routing.

This is an SRv6 instantiation of an SR-MPLS Binding SID [I-D.ietf-spring-segment-routing].

4.18. End.S: Endpoint in search of a target in table T

The "Endpoint in search of a target in Table T" function (End.S for short) is a variant of the End function.

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

```
1.   IF NH=SRH and SL = 0                                        ;; Ref1
2.      Send an ICMP parameter problem message                   ;; Ref2
3.      drop the packet
4.   ELSE IF match(last SID) in specified table T
5.      forward accordingly
6.   ELSE
7.      apply the End behavior
```

Ref1: By definition, an End.S SID cannot be the last SID, as the last SID is the targeted object.

Ref2: ICMP error is sent to the source address with error code (TBD by IANA) "SR Upper-layer Header Error" and pointer set to the NH.

The End.S function is required in information-centric networking (ICN) use-cases where the last SID in the SRv6 SID list represents a targeted object. If the identification of the object would require

more than 128 bits, then obvious customization of the End.S function may either use multiple SIDs or a TLV of the SR header to encode the searched object ID.

4.19. SR-aware application

Generally, any SR-aware application can be bound to an SRv6 SID. This application could represent anything from a small piece of code focused on topological/tenant function to a larger process focusing on higher-level applications (e.g. video compression, transcoding etc.).

The ways in which an SR-aware application binds itself on a local SID depends on the operating system. Let us consider an SR-aware application running on a Linux operating system. A possible approach is to associate an SRv6 SID to a target (virtual) interface, so that packets with IP DA corresponding to the SID will be sent to the target interface. In this approach, the SR-aware application can simply listen to all packets received on the interface.

A different approach for the SR-aware app is to listen to packets received with a specific SRv6 SID as IPv6 DA on a given transport port (i.e. corresponding to a TCP or UDP socket). In this case, the app can read the SRH information with a getsockopt Linux system call and can set the SRH information to be added to the outgoing packets with a setsocksopt system call.

4.20. Non SR-aware application

[I-D.xuclad-spring-sr-service-programming] defines a set of additional functions in order to enable non SR-aware applications to be associated with an SRv6 SID.

4.21. Flavours

We present the PSP and USP variants of the functions End, End.X and End.T. For each of these functions these variants can be enabled or disabled either individually or together.

4.21.1. PSP: Penultimate Segment Pop of the SRH

After the instruction 'update the IPv6 DA with SRH[SL]' is executed, the following instructions must be added:

```
1.   IF updated SL = 0 & PSP is TRUE
2.      pop the top SRH                                    ;; Ref1
```

Ref1: The received SRH had SL=1. When the last SID is written in the DA, the End, End.X and End.T functions with the PSP flavour pop the first (top-most) SRH. Subsequent stacked SRH's may be present but are not processed as part of the function.

4.21.2. USP: Ultimate Segment Pop of the SRH

We insert at the beginning of the pseudo-code the following instructions:

```
1.   IF NH=SRH & SL = 0 & USP=TRUE                              ;; Ref1
2.      pop the top SRH
3.      restart the function processing on the modified packet  ;; Ref2
```

Ref1: The next header is an SRH header

Ref2: Typically SL of the exposed SRH is > 0 and hence the restarting of the complete function would lead to decrement SL, update the IPv6 DA with SRH[SL], FIB lookup on updated DA and forward accordingly to the matched entry.

4.21.3. USD: Ultimate Segment Decapsulation

We insert at the beginning of the pseudo-code the following instructions:

```
1.   IF (NH=41) or (NH = SRH and SL = 0 and NNH = 41)
2.      pop the (outer) IPv6 header and its extension headers
3.      lookup the exposed inner IP DA and forward              ;; Ref1
4.      forward via the matched table entry
```

Ref1: In case that the USD flavor is applied on an End.X function, the packet is forwarded to the layer-3 adjancency bound to SID S without any lookup.

5. Transit behaviors

We define hereafter the set of basic transit behaviors. These behaviors are not bound to a SID and they correspond to source SR nodes or transit nodes [I-D.ietf-6man-segment-routing-header].

T	Transit behavior
T.Insert	Transit behavior with insertion of an SRv6 policy
T.Insert.Red	Transit behavior with reduced insert of an SRv6 policy
T.Encaps	Transit behavior with encapsulation in an SRv6 policy
T.Encaps.Red	Transit behavior with reduced encaps in an SRv6 policy
T.Encaps.L2	T.Encaps applied to received L2 frames
T.Encaps.L2.Red	T.Encaps.Red applied to received L2 frames

This list can be expanded in case any new functionality requires it.

5.1. T: Transit behavior

As per [RFC8200], if a node N receives a packet (A, S2)(S3, S2, S1; SL=2) and S2 is neither a local address nor a local SID of N then N forwards the packet without inspecting the SRH.

This means that N treats the following two packets with the same performance:

- (A, S2)

- (A, S2)(S3, S2, S1; SL=2)

A transit node does not need to count by default the amount of transit traffic with an SRH extension header. This accounting might be enabled as an optional behavior.

A transit node MUST include the outer flow label in its ECMP load-balancing hash [RFC6437].

5.2. T.Insert: Transit with insertion of an SRv6 Policy

Node N receives two packets P1=(A, B2) and P2=(A,B2)(B3, B2, B1; SL=1). B2 is neither a local address nor SID of N.

N steers the transit packets P1 and P2 into an SRv6 Policy with one SID list <S1, S2, S3>.

The "T.Insert" transit insertion behavior is defined as follows:

1. insert the SRH (B2, S3, S2, S1; SL=3) ;; Ref1, Ref1bis
2. set the IPv6 DA = S1
3. forward along the shortest path to S1

Ref1: The received IPv6 DA is placed as last SID of the inserted SRH.

Ref1bis: The SRH is inserted before any other IPv6 Routing Extension Header.

After the T.Insert behavior, P1 and P2 respectively look like:

- (A, S1) (B2, S3, S2, S1; SL=3)

- (A, S1) (B2, S3, S2, S1; SL=3) (B3, B2, B1; SL=1)

5.3. T.Insert.Red: Transit with reduced insertion

The T.Insert.Red behavior is an optimization of the T.Insert behavior. It is defined as follows:

1. insert the SRH (B2, S3, S2; SL=3)
2. set the IPv6 DA = S1
3. forward along the shortest path to S1

T.Insert.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the pushed SRH. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

After the T.Insert.Red behavior, P1 and P2 respectively look like:

- (A, S1) (B2, S3, S2; SL=3)

- (A, S1) (B2, S3, S2; SL=3) (B3, B2, B1; SL=1)

5.4. T.Encaps: Transit with encapsulation in an SRv6 Policy

Node N receives two packets P1=(A, B2) and P2=(A,B2)(B3, B2, B1; SL=1). B2 is neither a local address nor SID of N.

N steers the transit packets P1 and P2 into an SR Encapsulation Policy with a Source Address T and a Segment list <S1, S2, S3>.

The T.Encaps transit encapsulation behavior is defined as follows:

```
1.   push an IPv6 header with its own SRH (S3, S2, S1; SL=2)
2.   set outer IPv6 SA = T and outer IPv6 DA = S1
3.   set outer payload length, traffic class and flow label    ;; Ref1,2
4.   update the Next-Header value                              ;; Ref1
5.   decrement inner Hop Limit or TTL                          ;; Ref1
6.   forward along the shortest path to S1
```

After the T.Encaps behavior, P1 and P2 respectively look like:

- (T, S1) (S3, S2, S1; SL=2) (A, B2)

- (T, S1) (S3, S2, S1; SL=2) (A, B2) (B3, B2, B1; SL=1)

The T.Encaps behavior is valid for any kind of Layer-3 traffic. This behavior is commonly used for L3VPN with IPv4 and IPv6 deployments.

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

Ref 1: As described in [RFC2473] (Generic Packet Tunneling in IPv6 Specification)

Ref 2: As described in [RFC6437] (IPv6 Flow Label Specification)

5.5. T.Encaps.Red: Transit with reduced encapsulation

The T.Encaps.Red behavior is an optimization of the T.Encaps behavior. It is defined as follows:

```
1.   push an IPv6 header with its own SRH (S3, S2; SL=2)
2.   set outer IPv6 SA = T and outer IPv6 DA = S1
3.   set outer payload length, traffic class and flow label   ;; Ref1,2
4.   update the Next-Header value                             ;; Ref1
5.   decrement inner Hop Limit or TTL                         ;; Ref1
6.   forward along the shortest path to S1
```

Ref 1: As described in [RFC2473] (Generic Packet Tunneling in IPv6 Specification)

Ref 2: As described in [RFC6437] (IPv6 Flow Label Specification)

T.Encaps.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the SRH of the pushed IPv6 packet. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

After the T.Encaps.Red behavior, P1 and P2 respectively look like:

- (T, S1) (S3, S2; SL=2) (A, B2)

- (T, S1) (S3, S2; SL=2) (A, B2) (B3, B2, B1; SL=1)

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

5.6. T.Encaps.L2: Transit with encapsulation of L2 frames

While T.Encaps encapsulates the received IP packet, T.Encaps.L2 encapsulates the received L2 frame (i.e. the received ethernet header and its optional VLAN header is in the payload of the outer packet).

If the outer header is pushed without SRH, then the DA must be a SID of type End.DX2, End.DX2V, End.DT2U or End.DT2M and the next-header must be 59 (IPv6 NoNextHeader). The received Ethernet frame follows the IPv6 header and its extension headers.

Else, if the outer header is pushed with an SRH, then the last SID of the SRH must be of type End.DX2, End.DX2V, End.DT2U or End.DT2M and the next-header of the SRH must be 59 (IPv6 NoNextHeader). The received Ethernet frame follows the IPv6 header and its extension headers.

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

5.7. T.Encaps.L2.Red: Transit with reduced encaps of L2 frames

The T.Encaps.L2.Red behavior is an optimization of the T.Encaps.L2 behavior.

T.Encaps.L2.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the SRH of the pushed IPv6 packet. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

6. Operation

6.1. Counters

Any SRv6 capable node SHOULD implement the following set of combined counters (packets and bytes):

- CNT-1: Per entry of the "My SID Table", traffic that matched that SID and was processed correctly.

- CNT-2: Per SRv6 Policy, traffic steered into it and processed correctly.

Furthermore, an SRv6 capable node maintains an aggregate counter CNT-3 tracking the IPv6 traffic that was received with a destination address matching a local interface address that is not a locally instantiated SID and the next-header is SRH with SL>0. We remind that this traffic is dropped as an interface address is not a local SID by default. A SID must be explicitly instantiated.

6.2. Flow-based hash computation

When a flow-based selection within a set needs to be performed, the source address, the destination address and the flow-label MUST be included in the flow-based hash.

This occurs when the destination address is updated, a FIB lookup is performed and multiple ECMP paths exist to the updated destination address.

This occurs when End.X, End.DX4, or End.DX6 are bound to an array of adjacencies.

This occurs when the packet is steered in an SR policy whose selected path has multiple SID lists [I-D.filsfils-spring-segment-routing-policy].

6.3. OAM

[I-D.ali-spring-srv6-oam] defines the OAM behavior for SRv6. This includes the definition of the SRH Flag 'O-bit', as well as additional OAM Endpoint functions.

7. Basic security for intra-domain deployment

We use the following terminology:

An internal node is a node part of the domain of trust.

A border router is an internal node at the edge of the domain of trust.

An external interface is an interface of a border router towards another domain.

An internal interface is an interface entirely within the domain of trust.

The internal address space is the IP address block dedicated to internal interfaces.

An internal SID is a SID instantiated on an internal node.

The internal SID space is the IP address block dedicated to internal SIDs.

External traffic is traffic received from an external interface to the domain of trust.

Internal traffic is traffic that originates and ends within the domain of trust.

The purpose of this section is to document how a domain of trust can operate SRv6-based services for internal traffic while preventing any external traffic from accessing the internal SRv6-based services.

It is expected that future documents will detail enhanced security mechanisms for SRv6 (e.g. how to allow external traffic to leverage internal SRv6 services).

7.1. SEC-1

An SRv6 router MUST support an ACL on the external interface that drops any traffic with SA or DA in the internal SID space.

A provider would generally do this for its internal address space to prevent access to internal addresses and in order to prevent spoofing. The technique is extended to the local SID space.

The typical counters of an ACL are expected.

7.2. SEC-2

An SRv6 router MUST support an ACL with the following behavior:

1. IF (DA == LocalSID) && (SA != internal address or SID space)
2. drop

This prevents access to locally instantiated SIDs from outside the operator's infrastructure. Note that this ACL may not be enabled in all cases. For example, specific SIDs can be used to provide resources to devices that are outside of the operator's infrastructure.

The typical counters of an ACL are expected.

7.3. SEC-3

As per the End definition, an SRv6 router MUST only implement the End behavior on a local IPv6 address if that address has been explicitly enabled as an SRv6 SID.

This address may or may not be associated with an interface. This address may or may not be routed. The only thing that matters is that the local SID must be explicitly instantiated and explicitly bound to a function.

Packets received with destination address representing a local interface that has not been enabled as an SRv6 SID MUST be dropped.

8. Control Plane

In an SDN environment, one expects the controller to explicitly provision the SIDs and/or discover them as part of a service discovery function. Applications residing on top of the controller could then discover the required SIDs and combine them to form a distributed network program.

The concept of "SRv6 network programming" refers to the capability for an application to encode any complex program as a set of individual functions distributed through the network. Some functions relate to underlay SLA, others to overlay/tenant, others to complex applications residing in VM and containers.

The specification of the SRv6 control-plane is outside the scope of this document.

We limit ourselves to a few important observations.

8.1. IGP

The End, End.T and End.X SIDs express topological functions and hence are expected to be signaled in the IGP together with the flavours PSP, USP and USD[I-D.bashandy-isis-srv6-extensions].

The presence of SIDs in the IGP do not imply any routing semantics to the addresses represented by these SIDs. The routing reachability to an IPv6 address is solely governed by the classic, non-SID-related, IGP information. Routing is not governed neither influenced in any way by a SID advertisement in the IGP.

These three SIDs provide important topological functions for the IGP to build FRR/TI-LFA solution and for TE processes relying on IGP LSDB to build SR policies.

8.2. BGP-LS

BGP-LS is expected to be the key service discovery protocol. Every node is expected to advertise via BGP-LS its SRv6 capabilities (e.g. how many SIDs in can insert as part of an T.Insert behavior) and any locally instantiated SID [I-D.dawra-idr-bgpls-srv6-ext].

8.3. BGP IP/VPN/EVPN

The End.DX4, End.DX6, End.DT4, End.DT6, End.DT46, End.DX2, End.DX2V, End.DT2U and End.DT2M SIDs are expected to be signaled in BGP [I-D.dawra-idr-srv6-vpn].

8.4. Summary

The following table summarizes which SIDs are signaled in which signaling protocol.

	IGP	BGP-LS	BGP IP/VPN/EVPN
End (PSP, USP, USD)	X	X	
End.X (PSP, USP, USD)	X	X	
End.T (PSP, USP, USD)	X	X	
End.DX2		X	X
End.DX2V		X	X
End.DT2U		X	X
End.DT2M		X	X
End.DX6	X	X	X
End.DX4	X	X	X
End.DT6	X	X	X
End.DT4	X	X	X
End.DT46	X	X	X
End.B6.Insert		X	
End.B6.Insert.Red		X	
End.B6.Encaps		X	
End.B6.Encaps.Red		X	
End.B6.BM		X	
End.S		X	

Table 1: SRv6 locally instanted SIDs signaling

The following table summarizes which transit capabilities are signaled in which signaling protocol.

	IGP	BGP-LS	BGP IP/VPN/EVPN
T		X	
T.Insert	X	X	
T.Insert.Red	X	X	
T.Encaps	X	X	
T.Encaps.Red	X	X	
T.Encaps.L2		X	
T.Encaps.L2.Red		X	

Table 2: SRv6 transit behaviors signaling

The previous table describes generic capabilities. It does not describe specific instantiated SR policies.

For example, a BGP-LS advertisement of the T capability of node N would indicate that node N supports the basic transit behavior. The T.Insert behavior would describe the capability of node N to perform a T.Insert behavior, specifically it would describe how many SIDs could be inserted by N without significant performance degradation. Same for T.Encaps (the number is potentially lower as the overhead of the additional outer IP header is accounted).

The reader should also remember that any specific instantiated SR policy is always assigned a Binding SID. They should remember that BSIDs are advertised in BGP-LS as shown in Table 1. Hence, it is normal that Table 2 only focuses on the generic capabilities related to T.Insert and T.Encaps as Table 1 advertises the specific instantiated BSID properties.

9. IANA Considerations

This document requests the following new IANA registries:

- A new top-level registry "Segment-routing with IPv6 dataplane (SRv6) Parameters" to be created under IANA Protocol registries. This registry is being defined to serve as a top-level registry for keeping all other SRv6 sub-registries.

- A sub-registry "SRv6 Endpoint Behaviors" to be defined under top-level "Segment-routing with IPv6 dataplane (SRv6) Parameters" registry. This sub-registry maintains 16-bit identifiers for the SRv6 Endpoint behaviors. The range of the registry is 0-65535 (0x0000 - 0xFFFF) and has the following registration rules and allocation policies:

Range	Hex	Registration proceadure	Notes
0	0x0000	Reserved	Invalid
1-32767	0x0001-0x7FFF	IETF review	Draft Specifications
32768-49151	0x8000-0xBFFF	Reserved for experimental use	
49152-65534	0xC000-0xFFFE	Reserved for private use	
65535	0xFFFF	Reserved	Opaque

Table 3: SRv6 Endpoint Behaviors Registry

The initial registrations for the "Draft Specifications" portion of the sub-registry are as follows:

Value	Hex	Endpoint function	Reference
1	0x0001	End (no PSP, no USP)	[This.ID]
2	0x0002	End with PSP	[This.ID]
3	0x0003	End with USP	[This.ID]
4	0x0004	End with PSP&USP	[This.ID]
5	0x0005	End.X (no PSP, no USP)	[This.ID]
6	0x0006	End.X with PSP	[This.ID]
7	0x0007	End.X with USP	[This.ID]
8	0x0008	End.X with PSP&USP	[This.ID]
9	0x0009	End.T (no PSP, no USP)	[This.ID]
10	0x000A	End.T with PSP	[This.ID]
11	0x000B	End.T with USP	[This.ID]
12	0x000C	End.T with PSP&USP	[This.ID]
13	0x000D	End.B6	[This.ID]
14	0x000E	End.B6.Encaps	[This.ID]
15	0x000F	End.BM	[This.ID]
16	0x0010	End.DX6	[This.ID]
17	0x0011	End.DX4	[This.ID]
18	0x0012	End.DT6	[This.ID]
19	0x0013	End.DT4	[This.ID]
20	0x0014	End.DT46	[This.ID]
21	0x0015	End.DX2	[This.ID]
22	0x0016	End.DX2V	[This.ID]
23	0x0017	End.DT2U	[This.ID]
24	0x0018	End.DT2M	[This.ID]
25	0x0019	End.S	[This.ID]
26	0x001A	End.B6.Red	[This.ID]
27	0x001B	End.B6.Encaps.Red	[This.ID]
28	0x001C	End with USD	[This.ID]
29	0x001D	End with PSP&USD	[This.ID]
30	0x001E	End with USP&USD	[This.ID]
31	0x001F	End with PSP, USP & USD	[This.ID]
32	0x0020	End.X with USD	[This.ID]
33	0x0021	End.X with PSP&USD	[This.ID]
34	0x0022	End.X with USP&USD	[This.ID]
35	0x0023	End.X with PSP, USP & USD	[This.ID]
36	0x0024	End.T with USD	[This.ID]
37	0x0025	End.T with PSP&USD	[This.ID]
38	0x0026	End.T with USP&USD	[This.ID]
39	0x0027	End.T with PSP, USP & USD	[This.ID]

Table 4: IETF - SRv6 Endpoint Behaviors

10. Work in progress

We are working on a extension of this document to provide Yang modelling for all the functionality described in this document. This work is ongoing in [I-D.raza-spring-srv6-yang].

11. Acknowledgements

The authors would like to acknowledge Stefano Previdi, Dave Barach, Mark Townsley, Peter Psenak, Thierry Couture, Kris Michielsen, Paul Wells, Robert Hanzl, Dan Ye, Gaurav Dawra, Faisal Iqbal, Jaganbabu Rajamanickam, David Toscano, Asif Islam, Jianda Liu, Yunpeng Zhang, Jiaoming Li, Narendra A.K, Mike Mc Gourty, Bhupendra Yadav, Sherif Toulan, Satish Damodaran, John Bettink, Kishore Nandyala Veera Venk, Jisu Bhattacharya and Saleem Hafeez.

12. Contributors

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Dirk Steinberg
Steinberg Consulting
Germany

Email: dws@dirksteinberg.de

Robert Raszuk
Bloomberg LP
United States of America

Email: robert@raszuk.net

Bruno Decraene
Orange
Frence

Email: bruno.decraene@orange.com

Bart Peirens
Proximus
Belgium

Email: bart.peirens@proximus.com

Hani Elmalky
Ericsson
United States of America

Email: hani.elmalky@gmail.com

Prem Jonnalagadda
Barefoot Networks
United States of America

Email: prem@barefootnetworks.com

Milad Sharif
Barefoot Networks
United States of America

Email: msharif@barefootnetworks.com

David Lebrun
Google
Belgium

Email: dlebrun@google.com

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

Ahmed AbdelSalam
Gran Sasso Science Institute
Italy

Email: ahmed.abdelsalam@gssi.it

Gaurav Naik
Drexel University
United States of America

Email: gn@drexel.edu

Arthi Ayyangar
Arista
United States of America

Email: arthi@arista.com

Satish Mynam
Innovium Inc.
United States of America

Email: smynam@innovium.com

Wim Henderickx
Nokia
Belgium

Email: wim.henderickx@nokia.com

Shaowen Ma
Juniper
Singapore

Email: mashao@juniper.net

Ahmed Bashandy
Individual
United States of America

Email: abashandy.ietf@gmail.com

Francois Clad
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Kamran Raza
Cisco Systems, Inc.
Canada

Email: skraza@cisco.com

Darren Dukes
Cisco Systems, Inc.
Canada

Email: ddukes@cisco.com

Patrice Brissete
Cisco Systems, Inc.
Canada

Email: pbrisset@cisco.com

Zafar Ali
Cisco Systems, Inc.
United States of America

Email: zali@cisco.com

13. References

13.1. Normative References

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-16 (work in progress), February 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>.

[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>.

13.2. Informative References

[I-D.ali-spring-srv6-oam]
Ali, Z., Filsfils, C., Kumar, N., Pignataro, C., faiqbal@cisco.com, f., Gandhi, R., Leddy, J., Matsushima, S., Raszuk, R., daniel.voyer@bell.ca, d., Dawra, G., Peirens, B., Chen, M., and G. Naik, "Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)", draft-ali-spring-srv6-oam-02 (work in progress), October 2018.

[I-D.bashandy-isis-srv6-extensions]
Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Routing over IPv6 Dataplane", draft-bashandy-isis-srv6-extensions-04 (work in progress), October 2018.

[I-D.dawra-idr-bgpls-srv6-ext]
Dawra, G., Filsfils, C., Talaulikar, K., Chen, M., daniel.bernier@bell.ca, d., Uttaro, J., Decraene, B., and H. Elmalky, "BGP Link State extensions for IPv6 Segment Routing(SRv6)", draft-dawra-idr-bgpls-srv6-ext-04 (work in progress), September 2018.

[I-D.dawra-idr-srv6-vpn]
Dawra, G., Filsfils, C., Dukes, D., Brissette, P., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Decraene, B., Matsushima, S., and S. Zhuang, "BGP Signaling for SRv6 based Services.", draft-dawra-idr-srv6-vpn-05 (work in progress), October 2018.

[I-D.filsfils-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Hegde, S., daniel.voyer@bell.ca, d., Lin, S., bogdanov@google.com, b., Krol, P., Horneffer, M., Steinberg, D., Decraene, B., Litkowski, S., Mattes, P., Ali, Z., Talaulikar, K., Liste, J., Clad, F., and K. Raza, "Segment Routing Policy Architecture", draft-filsfils-spring-segment-routing-policy-06 (work in progress), May 2018.

[I-D.filsfils-spring-srv6-net-pgm-illustration]
Filsfils, C., Camarillo, P., Li, Z., Matsushima, S., Decraene, B., Steinberg, D., Lebrun, D., Raszuk, R., and J. Leddy, "Illustrations for SRv6 Network Programming", draft-filsfils-spring-srv6-net-pgm-illustration-00 (work in progress), February 2019.

[I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H., and M. Chen, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-11 (work in progress), October 2018.

[I-D.ietf-idr-te-lsp-distribution]
Previdi, S., Talaulikar, K., Dong, J., Chen, M., Gredler, H., and J. Tantsura, "Distribution of Traffic Engineering (TE) Policies and State using BGP-LS", draft-ietf-idr-te-lsp-distribution-09 (work in progress), June 2018.

[I-D.ietf-isis-l2bundles]
Ginsberg, L., Bashandy, A., Filsfils, C., Nanduri, M., and E. Aries, "Advertising L2 Bundle Member Link Attributes in IS-IS", draft-ietf-isis-l2bundles-07 (work in progress), May 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[I-D.raza-spring-srv6-yang]
Raza, K., Rajamanickam, J., Liu, X., Hu, Z., Hussain, I., Shah, H., daniel.voyer@bell.ca, d., Elmalky, H., Matsushima, S., Horiba, K., and A. Abdelsalam, "YANG Data Model for SRv6 Base and Static", draft-raza-spring-srv6-yang-02 (work in progress), October 2018.

[I-D.xuclad-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca, d., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", draft-xuclad-spring-sr-service-programming-01 (work in progress), October 2018.

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

[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>.

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

[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>.

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Pablo Camarillo Garvia (editor)
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

John Leddy
Comcast
United States of America

Email: john_leddy@cable.comcast.com

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Satoru Matsushima
SoftBank
1-9-1,Higashi-Shimbashi,Minato-Ku
Tokyo 105-7322
Japan

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

Zhenbin Li
Huawei Technologies
China

Email: lizhenbin@huawei.com

SPRING Working Group
Internet-Draft
Intended Status: Informational
Expires: December 11, 2018

R. Gandhi, Ed.
C. Filsfils
S. Soni
Cisco Systems, Inc.
D. Voyer
Bell Canada
S. Salsano
Universita di Roma "Tor Vergata"
P. Ventre
CNIT

June 9, 2018

Performance Measurement in Segment Routing Networks with MPLS Data Plane draft-gandhi-spring-sr-mpls-pm-01

Abstract

RFC 6374 specifies protocol mechanisms to enable the efficient and accurate measurement of packet loss, one-way and two-way delay, as well as related metrics such as delay variation in MPLS networks. This document reviews how these mechanisms can be used for Delay and Loss Performance Measurements (PM) in Segment Routing (SR) networks with MPLS data plane (SR-MPLS), for both SR links and end-to-end SR Policies. The measured performance values for SR links are used to compute extended metrics for delay and loss and are advertised in the network.

Status of This Memo

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

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

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

Copyright Notice

Table of Contents

1. Introduction

Service provider's ability to satisfy Service Level Agreements (SLAs) depend on the ability to measure and monitor performance metrics for packet loss and one-way and two-way delay, as well as related metrics such as delay variation. The ability to monitor these performance metrics also provides operators with greater visibility into the performance characteristics of their networks, thereby facilitating planning, troubleshooting, and network performance evaluation.

[RFC6374] specifies protocol mechanisms to enable the efficient and accurate measurement of these performance metrics in MPLS networks. The One-Way Active Measurement Protocol (OWAMP) defined in [RFC4656] and Two-Way Active Measurement Protocol (TWAMP) defined in [RFC5357] provide capabilities for the measurement of various performance metrics in IP networks. However, mechanisms defined in [RFC6374] are more suitable for Segment Routing (SR) when using MPLS data plane (SR-MPLS). In addition, [RFC6374] also supports IEEE 1588 timestamps [IEEE1588] and "direct mode" Loss Measurement (LM), which are required in SR networks.

[RFC7876] specifies the procedures to be used when sending and processing out-of-band performance measurement probe replies over an UDP return path when receiving RFC 6374 based probe queries. These procedures can be used to send out-of-band PM replies for both SR links and SR Policies for one-way measurement.

This document reviews how mechanisms defined in [RFC6374] can be used for Delay and Loss Performance Measurements (PM) in SR networks with MPLS data plane, for both SR links and end-to-end SR Policies. The measured performance values for SR links are used to compute extended metrics for delay and loss and are advertised in the network.

2. Conventions Used in This Document

2.1. Abbreviations

ACH: Associated Channel Header.

DFLag: Data Format Flag.

DM: Delay Measurement.

ECMP: Equal Cost Multi-Path.

G-ACh: Generic Associated Channel (G-ACh)

GAL: Generic Associated Channel (G-ACh) Label

LM: Loss Measurement.

MPLS: Multiprotocol Label Switching.

PM: Performance Measurement.

PTP: Precision Time Protocol.

SID: Segment ID.

SR: Segment Routing.

TC: Traffic Class.

URO: UDP Return Object.

2.2. Reference Topology

In the reference topology shown in Figure 1, the querier node R1 initiates a performance measurement probe query and the responder node R5 sends a probe response for the query message received. The probe response is sent to the querier node R1. The nodes R1 and R5 may be directly connected via a link enabled with Segment Routing or there exists an SR Policy [I-D.spring-segment-routing-policy] on node R1 with destination to node R5.

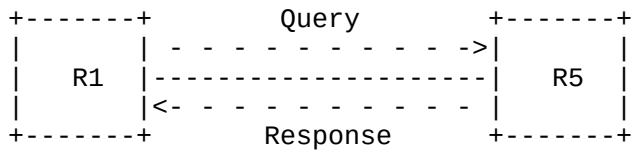


Figure 1: Reference Topology

Both delay and loss performance measurement is performed for the traffic traversing between node R1 and node R5. One-way delay and two-way delay measurements are defined in Section 2.4 of [RFC6374]. Transmit and Receive packet loss measurements are defined in Section 2.2 of [RFC6374]. One-way loss measurement provides transmit packet loss whereas two-way loss measurement provides both transmit and receive packet loss.

3. Probe Query and Response Packets

3.1. Probe Packet Header for SR-MPLS Policies

As described in Section 2.9.1 of [RFC6374], MPLS PM probe query and response messages flow over the MPLS Generic Associated Channel (G-ACh). A probe packet for an end-to-end measurement for SR Policy contains SR-MPLS label stack [I-D.spring-segment-routing-policy], with the G-ACh Label (GAL) at the bottom of the stack. The GAL is followed by an Associated Channel Header (ACH), which identifies the message type and the message payload following the ACH as shown in Figure 2.

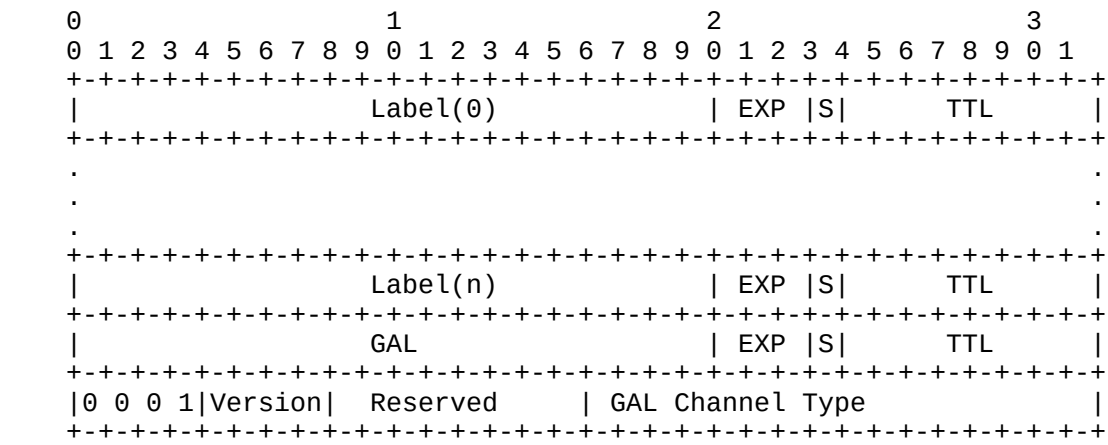
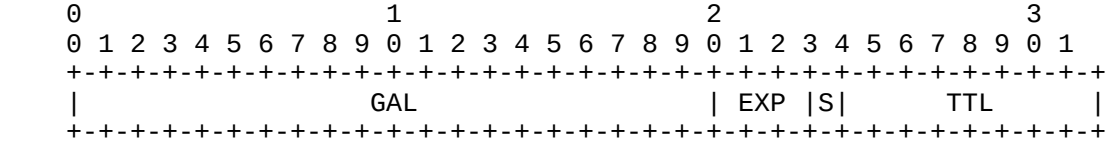


Figure 2: Probe Packet Header for an End-to-end SR-MPLS Policy

The SR-MPLS label stack can be empty to indicate Implicit NULL label case.

3.2. Probe Packet Header for SR-MPLS Links

As described in Section 2.9.1 of [RFC6374], MPLS PM probe query and response messages flow over the MPLS Generic Associated Channel (G-ACh). A probe packet for SR-MPLS links contains G-ACh Label (GAL). The GAL is followed by an Associated Channel Header (ACH), which identifies the message type, and the message payload following the ACH as shown in Figure 3.



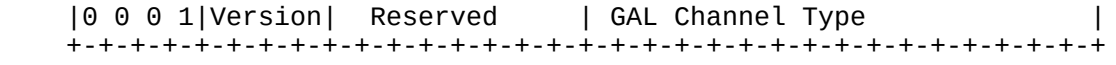


Figure 3: Probe Packet Header for an SR-MPLS Link

3.3. Probe Response Message for SR-MPLS Links and Policies

3.3.1. One-way Measurement Probe Response Message

For one-way performance measurement [RFC7679], the PM querier node can receive "out-of-band" probe replies by properly setting the UDP Return Object (URO) TLV in the probe query message. The URO TLV (Type=131) is defined in [RFC7876] and includes the UDP-Destination-Port and IP Address. In particular, if the querier sets its own IP address in the URO TLV, the probe response is sent back by the responder node to the querier node.

3.3.2. Two-way Measurement Probe Response Message

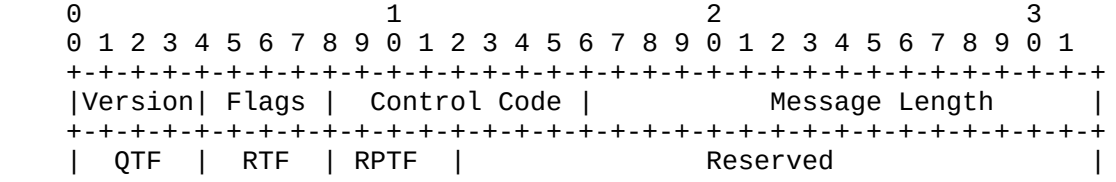
For two-way performance measurement [RFC6374], when using a bidirectional channel, the probe response message is sent back to the querier node in-band on the reverse direction SR Link or SR Policy using a message with format similar to their probe query message. In this case, the "control code" in the probe query message is set to "in-band response requested".

4. Performance Delay Measurement

4.1. Delay Measurement Message Format

As defined in [RFC6374], MPLS DM probe query and response messages use Associated Channel Header (ACH) (value 0x000C for delay measurement) [RFC6374], which identifies the message type, and the message payload following the ACH. For both SR links and end-to-end measurement for SR Policies, the same MPLS DM ACH value is used.

The DM message payload as defined in [RFC6374] is used for SR-MPLS delay measurement, for both SR links and end-to-end SR Policies. The DM message payload format is defined as following in [RFC6374]:



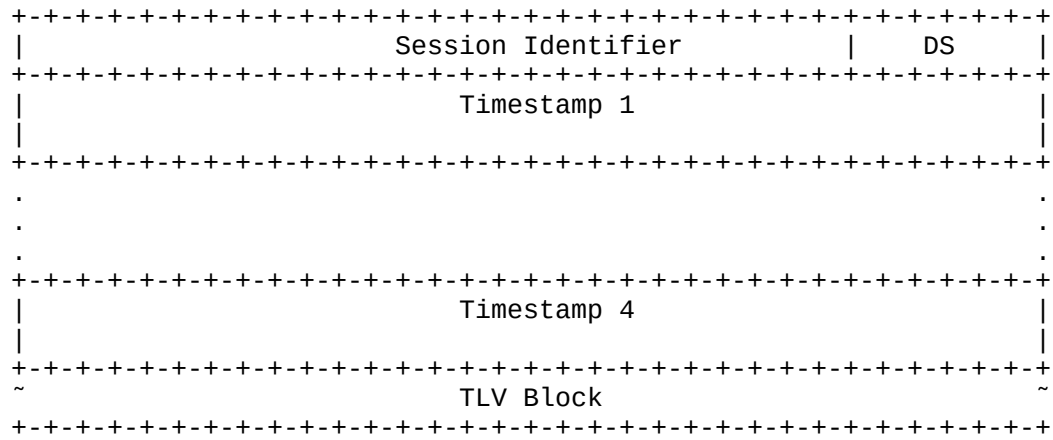


Figure 4: Delay Measurement Message Payload Format

The meanings of the fields are summarized in the following table, see [RFC6374] for details.

Field	Meaning
Version	Protocol version
Flags	Message control flags
Control Code	Code identifying the query or response type
QTF	Querier timestamp format (see Section 3.4 of [RFC6374])
RTF	Responder timestamp format (see Section 3.4 of [RFC6374])
RPTF	Responder’s preferred timestamp format
Reserved	Reserved for future specification
Session Identifier	Set arbitrarily by the querier
Differentiated Services (DS) Field	Differentiated Services Code Point (DSCP) being measured
Timestamp 1-4	64-bit timestamp values (see Section 3.4 of [RFC6374])
TLV Block	Optional block of Type-Length-Value fields

4.2. Timestamp

The Section 3.4 of [RFC6374] defines timestamp format that can be used for delay measurement. The IEEE 1588 Precision Time Protocol (PTP) timestamp format [IEEE1588] is used by default as described in Appendix A of [RFC6374], but it may require hardware support. As an alternative, Network Time Protocol (NTP) timestamp format can also be used [RFC6374].

Note that for one-way delay measurement, clock synchronization between the querier and responder nodes using methods detailed in [RFC6374] is required. The two-way delay measurement does not require clock to be synchronized between the querier and responder nodes.

5. Performance Loss Measurement

The LM protocol can perform two distinct kinds of loss measurement as described in Section 2.9.8 of [RFC6374].

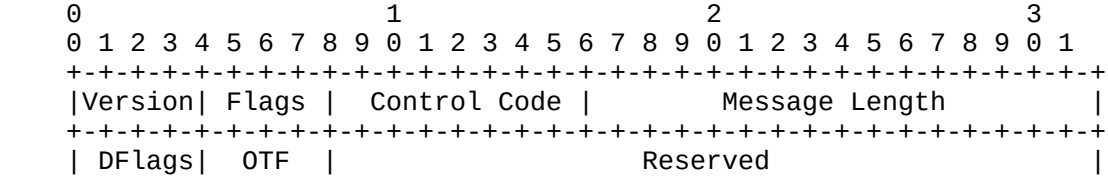
- In inferred mode, LM will measure the loss of specially generated test messages in order to infer the approximate data plane loss level. Inferred mode LM provides only approximate loss accounting.

- In direct mode, LM will directly measure data plane packet loss. Direct mode LM provides perfect loss accounting, but may require hardware support.

5.1. Loss Measurement Message Format

As defined in [RFC6374], MPLS LM probe query and response messages use Associated Channel Header (ACH) (value 0x000A for direct loss measurement or value 0x000B for inferred loss measurement), which identifies the message type, and the message payload following the ACH. For both SR links and end-to-end measurement for SR Policies, the same MPLS LM ACH value is used.

The LM message payload as defined in [RFC6374] is used for SR-MPLS loss measurement, for both SR links and end-to-end SR Policies. The LM message payload format is defined as following in [RFC6374]:



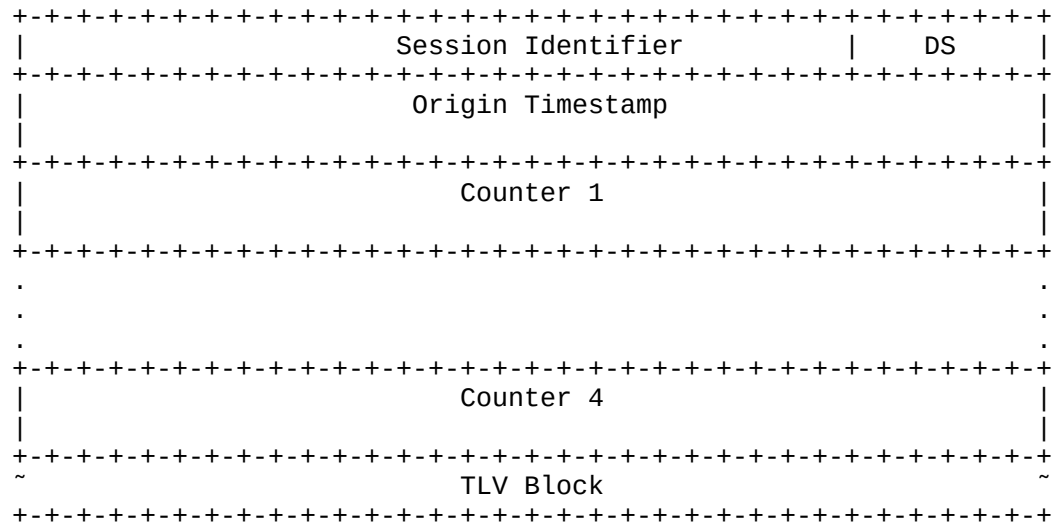


Figure 5: Loss Measurement Message Payload Format

The meanings of the fields are summarized in the following table, see [RFC6374] for details.

Field	Meaning
Version	Protocol version
Flags	Message control flags
Control Code	Code identifying the query or response type
Message Length	Total length of this message in bytes
Data Format Flags (DFlags)	Flags specifying the format of message data
Origin Timestamp Format (OTF)	Format of the Origin Timestamp field
Reserved	Reserved for future specification
Session Identifier	Set arbitrarily by the querier
Differentiated Services (DS) Field	Differentiated Services Code Point (DSCP) being measured
Origin Timestamp	64-bit field for query message transmission

	timestamp
Counter 1-4	64-bit fields for LM counter values
TLV Block	Optional block of Type-Length-Value fields

6. SR Link Metrics Advertisements

Performance metrics for SR link delay and loss computed using the performance measurement procedures reviewed in this document can be advertised in the routing domain as follows:

- For OSPF, ISIS, and BGP-LS, protocol extensions defined in [RFC7471], [RFC7810] [I-D.lsr-isis-rfc7810bis], and [I-D.idr-te-pm-bgp] are used, respectively for advertising the extended link metrics in the network.
- The extended link delay metrics advertised are minimum-delay, maximum-delay, average-delay and delay-variance for one-way.
- The delay-variance is computed as specified in Section 4.2 of [RFC5481].
- The one-way delay metrics can be computed using two-way measurement by dividing the measured delay values by 2.
- The extended link loss metric advertised is one-way percentage packet loss.

7. Security Considerations

This document reviews the procedures for performance measurement for SR-MPLS networks, for both SR-MPLS links and end-to-end SR-MPLS Policies using the mechanisms defined in [RFC6374]. This document does not introduce any additional security considerations other than those covered in [RFC6374].

8. IANA Considerations

This document does not require any IANA actions.

9. References

9.1. Normative References

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS networks', RFC 6374, September 2011.

[RFC7876] Bryant, S., Sivabalan, S., and Soni, S., "UDP Return Path for Packet Loss and Delay Measurement for MPLS Networks", RFC 7876, July 2016.

9.2. Informative References

[IEEE1588] IEEE, "1588-2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", March 2008.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protoco (OWAMP)", RFC 4656, September 2006.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, March 2009.

[RFC7679] Almes, G., et al., "A One-Way Delay Metric for IP Performance Metrics (IPPM)', RFC 7679, January 2016.

[RFC7471] Giacalone, S., et al., "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, March 2015.

[RFC7810] Previdi, S., et al., "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 7810, May 2016.

[I-D.lsr-isis-rfc7810bis] Ginsberg, L., et al., "IS-IS Traffic Engineering (TE) Metric Extensions", draft-ietf-lsr-isis-rfc7810bis, work in progress.

[I-D.idr-te-pm-bgp] Ginsberg, L. Ed., et al., "BGP-LS Advertisement of IGP Traffic Engineering Performance Metric Extensions", draft-ietf-idr-te-pm-bgp, work in progress.

[I-D.spring-segment-routing-policy] Filsfils, C., et al., "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy, work in progress.

Acknowledgments

To be added.

Contributors

Patrick Khordoc
Cisco Systems, Inc.
Email: pkhordoc@cisco.com

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

Daniel Bernier
Bell Canada
Email: daniel.bernier@bell.ca

Authors' Addresses

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

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

Sagar Soni
Cisco Systems, Inc.
Email: sagsoni@cisco.com

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

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

Pier Luigi Ventre
CNIT
Italy
Email: pierluigi.ventre@cnit.it

SPRING Working Group
Internet-Draft
Intended Status: Informational
Expires: March 18, 2019

R. Gandhi, Ed.
C. Filsfils
Cisco Systems, Inc.
D. Voyer
Bell Canada
S. Salsano
Universita di Roma "Tor Vergata"
P. L. Ventre
CNIT
M. Chen
Huawei

September 14, 2018

Performance Measurement in Segment Routing Networks with MPLS Data Plane draft-gandhi-spring-sr-mpls-pm-03

Abstract

RFC 6374 specifies protocol mechanisms to enable the efficient and accurate measurement of packet loss, one-way and two-way delay, as well as related metrics such as delay variation in MPLS networks using probe messages. This document reviews how these mechanisms can be used for Delay and Loss Performance Measurements (PM) in Segment Routing (SR) networks with MPLS data plane (SR-MPLS), for both SR links and end-to-end SR Policies. The performance measurements for SR links are used to compute extended Traffic Engineering (TE) metrics for delay and loss and are advertised in the network using routing protocol extensions.

Status of This Memo

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

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

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

Copyright Notice

Table of Contents

1. Introduction

Service provider's ability to satisfy Service Level Agreements (SLAs) depend on the ability to measure and monitor performance metrics for packet loss and one-way and two-way delay, as well as related metrics such as delay variation. The ability to monitor these performance metrics also provides operators with greater visibility into the performance characteristics of their networks, thereby facilitating planning, troubleshooting, and network performance evaluation.

[RFC6374] specifies protocol mechanisms to enable the efficient and accurate measurement of performance metrics in MPLS networks using probe messages. The One-Way Active Measurement Protocol (OWAMP) defined in [RFC4656] and Two-Way Active Measurement Protocol (TWAMP) defined in [RFC5357] provide capabilities for the measurement of various performance metrics in IP networks. However, mechanisms defined in [RFC6374] are more suitable for Segment Routing (SR) when using MPLS data plane (SR-MPLS). The [RFC6374] also supports IEEE 1588 timestamps [IEEE1588] and "direct mode" Loss Measurement (LM), which are required in SR networks.

[RFC7876] specifies the procedures to be used when sending and processing out-of-band performance measurement probe replies over an UDP return path when receiving RFC 6374 based probe queries. These procedures can be used to send out-of-band PM replies for both SR links and SR Policies [I-D.spring-segment-routing-policy] for one-way measurement.

This document reviews how probe based mechanisms defined in [RFC6374] can be used for Delay and Loss Performance Measurements (PM) in SR networks with MPLS data plane, for both SR links and end-to-end SR Policies. The performance measurements for SR links are used to compute extended Traffic Engineering (TE) metrics for delay and loss and are advertised in the network using routing protocol extensions.

2. Conventions Used in This Document

2.1. Abbreviations

ACH: Associated Channel Header.

DFLag: Data Format Flag.

DM: Delay Measurement.

ECMP: Equal Cost Multi-Path.

G-ACh: Generic Associated Channel (G-ACh).

GAL: Generic Associated Channel (G-ACh) Label.

LM: Loss Measurement.

MPLS: Multiprotocol Label Switching.

NTP: Network Time Protocol.

PM: Performance Measurement.

PTP: Precision Time Protocol.

SID: Segment ID.

SL: Segment List.

SR: Segment Routing.

SR-MPLS: Segment Routing with MPLS data plane.

TC: Traffic Class.

TE: Traffic Engineering.

URO: UDP Return Object.

2.2. Reference Topology

In the reference topology shown in Figure 1, the querier node R1 initiates a performance measurement probe query and the responder node R5 sends a probe response for the query message received. The probe response is typically sent to the querier node R1. The nodes R1 and R5 may be directly connected via a link enabled with Segment Routing or there exists a Point-to-Point (P2P) SR Policy [I-D.spring-segment-routing-policy] on node R1 with destination to node R5. In case of Point-to-Multipoint (P2MP), SR Policy originating from source node R1 may terminate on multiple destination leaf nodes [I-D.spring-sr-p2mp-policy].

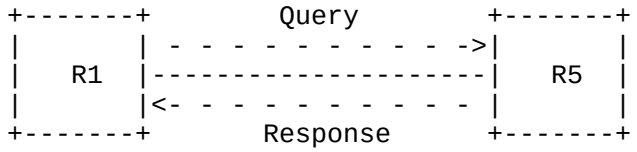


Figure 1: Reference Topology

Both delay and loss performance measurement is performed in-band for the traffic traversing between node R1 and node R5. One-way delay and two-way delay measurements are defined in Section 2.4 of [RFC6374]. Transmit and Receive packet loss measurements are defined in Section 2.2 and Section 2.6 of [RFC6374]. One-way loss measurement provides receive packet loss whereas two-way loss measurement provides both transmit and receive packet loss.

3. Probe Query and Response Packets

3.1. Probe Packet Header for SR-MPLS Policies

As described in Section 2.9.1 of [RFC6374], MPLS PM probe query and response messages flow over the MPLS Generic Associated Channel (G-ACh). A probe packet for an end-to-end measurement for SR Policy contains SR-MPLS label stack [I-D.spring-segment-routing-policy], with the G-ACh Label (GAL) at the bottom of the stack. The GAL is followed by an Associated Channel Header (ACH), which identifies the message type and the message payload following the ACH as shown in Figure 2.

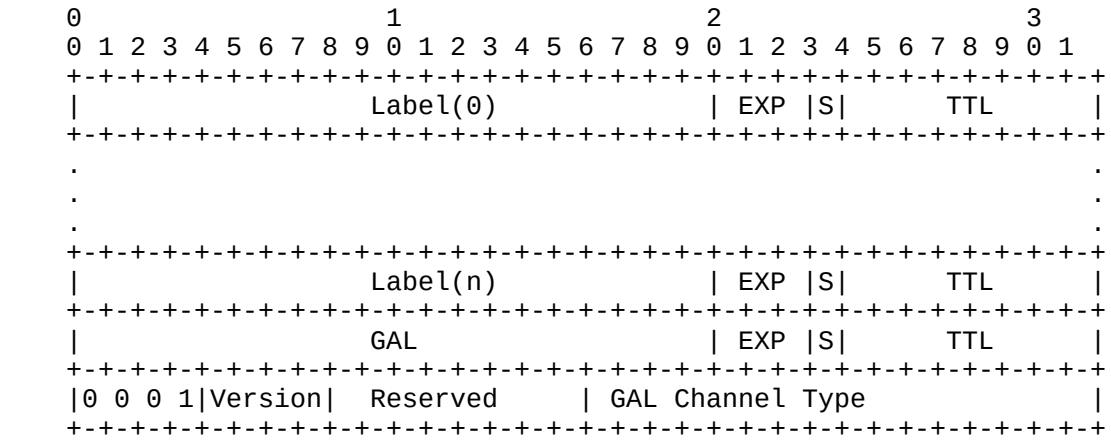


Figure 2: Probe Packet Header for an End-to-end SR-MPLS Policy

The SR-MPLS label stack can be empty to indicate Implicit NULL label case.

3.2. Probe Packet Header for SR-MPLS Links

As described in Section 2.9.1 of [RFC6374], MPLS PM probe query and

response messages flow over the MPLS Generic Associated Channel (G-ACh). A probe packet for SR-MPLS links contains G-ACh Label (GAL). The GAL is followed by an Associated Channel Header (ACH), which identifies the message type, and the message payload following the ACH as shown in Figure 3.

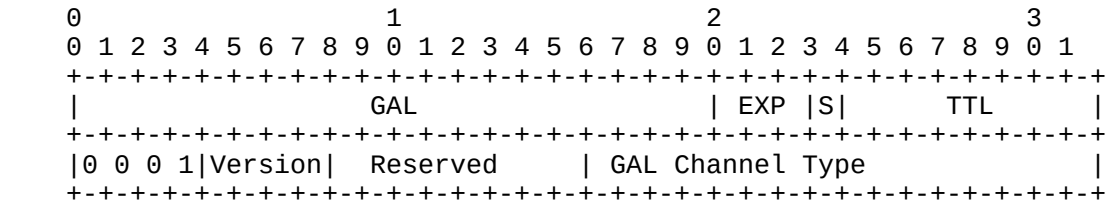


Figure 3: Probe Packet Header for an SR-MPLS Link

3.3. Probe Response Message for SR-MPLS Links and Policies

3.3.1. One-way Measurement Probe Response Message

For one-way performance measurement [RFC7679], the PM querier node can receive "out-of-band" probe replies by properly setting the UDP Return Object (URO) TLV in the probe query message. The URO TLV (Type=131) is defined in [RFC7876] and includes the UDP-Destination-Port and IP Address. In particular, if the querier sets its own IP address in the URO TLV, the probe response is sent back by the responder node to the querier node. In addition, the "control code" in the probe query message is set to "out-of-band response requested". The "Source Address" TLV (Type 130), and "Return Address" TLV (Type 1), if present in the probe query message, are not used to send probe response message.

3.3.2. Two-way Measurement Probe Response Message

For two-way performance measurement [RFC6374], when using a bidirectional channel, the probe response message is sent back to the querier node in-band on the reverse direction SR Link or SR Policy using a message with format similar to their probe query message. In this case, the "control code" in the probe query message is set to "in-band response requested".

A path segment identifier [I-D.spring-mpls-path-segment] [I-D.pce-sr-path-segment] of the forward SR Policy can be used to find the reverse SR Policy to send the probe response message.

4. Performance Delay Measurement

4.1. Delay Measurement Message Format

As defined in [RFC6374], MPLS DM probe query and response messages use Associated Channel Header (ACH) (value 0x000C for delay measurement) [RFC6374], which identifies the message type, and the message payload following the ACH. For both SR links and end-to-end measurement for SR Policies, the same MPLS DM ACH value is used.

The DM message payload as defined in [RFC6374] is used for SR-MPLS delay measurement, for both SR links and end-to-end SR Policies. The DM message payload format is defined as following in [RFC6374]:

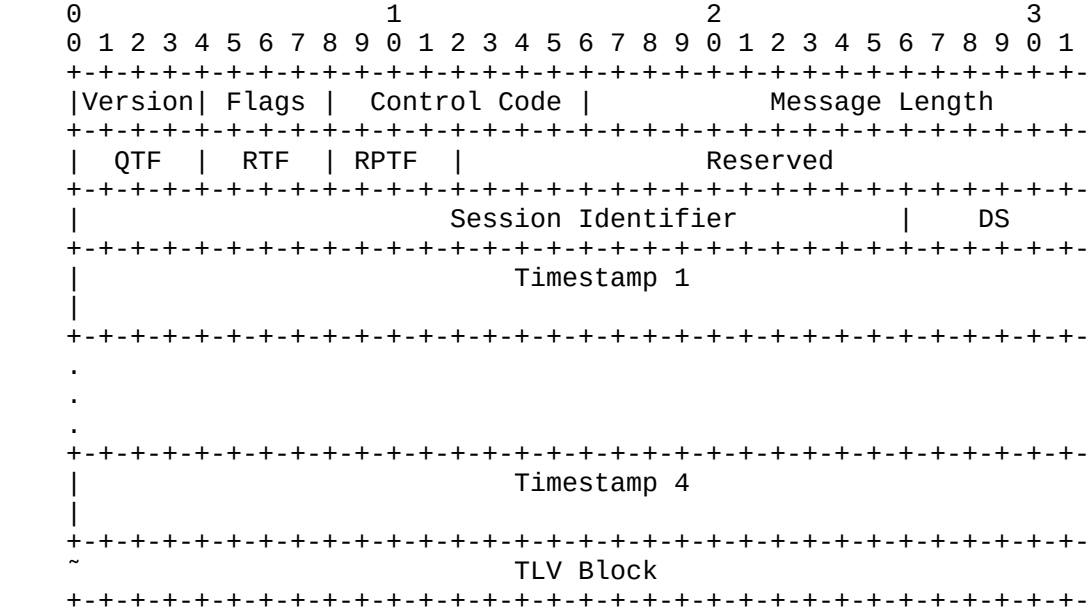


Figure 4: Delay Measurement Message Payload Format

The meanings of the fields are summarized in the following table, see [RFC6374] for details.

Field	Meaning
Version	Protocol version
Flags	Message control flags
Control Code	Code identifying the query or response type
QTF	Querier timestamp format

	(see Section 3.4 of [RFC6374])
RTF	Responder timestamp format (see Section 3.4 of [RFC6374])
RPTF	Responder's preferred timestamp format
Reserved	Reserved for future specification
Session Identifier	Set arbitrarily by the querier
Differentiated Services (DS) Field	Differentiated Services Code Point (DSCP) being measured
Timestamp 1-4	64-bit timestamp values (see Section 3.4 of [RFC6374])
TLV Block	Optional block of Type-Length-Value fields

4.2. Timestamps

The Section 3.4 of [RFC6374] defines timestamp format that can be used for delay measurement. The IEEE 1588 Precision Time Protocol (PTP) timestamp format [IEEE1588] is used by default as described in Appendix A of [RFC6374], but it may require hardware support. As an alternative, Network Time Protocol (NTP) timestamp format can also be used [RFC6374].

Note that for one-way delay measurement, clock synchronization between the querier and responder nodes using the methods detailed in [RFC6374] is required. The two-way delay measurement does not require clock synchronization between the querier and responder nodes.

5. Performance Loss Measurement

The LM protocol can perform two distinct kinds of loss measurement as described in Section 2.9.8 of [RFC6374].

- In inferred mode, LM will measure the loss of specially generated test messages in order to infer the approximate data plane loss level. Inferred mode LM provides only approximate loss accounting.

- In direct mode, LM will directly measure data plane packet loss. Direct mode LM provides perfect loss accounting, but may require

hardware support.

For both of these modes of LM, path segment identifier [I-D.spring-mpls-path-segment] [I-D.pce-sr-path-segment] is required for accounting received traffic on the egress node of the SR-MPLS Policy.

5.1. Loss Measurement Message Format

As defined in [RFC6374], MPLS LM probe query and response messages use Associated Channel Header (ACH) (value 0x000A for direct loss measurement or value 0x000B for inferred loss measurement), which identifies the message type, and the message payload following the ACH. For both SR links and end-to-end measurement for SR Policies, the same MPLS LM ACH value is used.

The LM message payload as defined in [RFC6374] is used for SR-MPLS loss measurement, for both SR links and end-to-end SR Policies. The LM message payload format is defined as following in [RFC6374]:

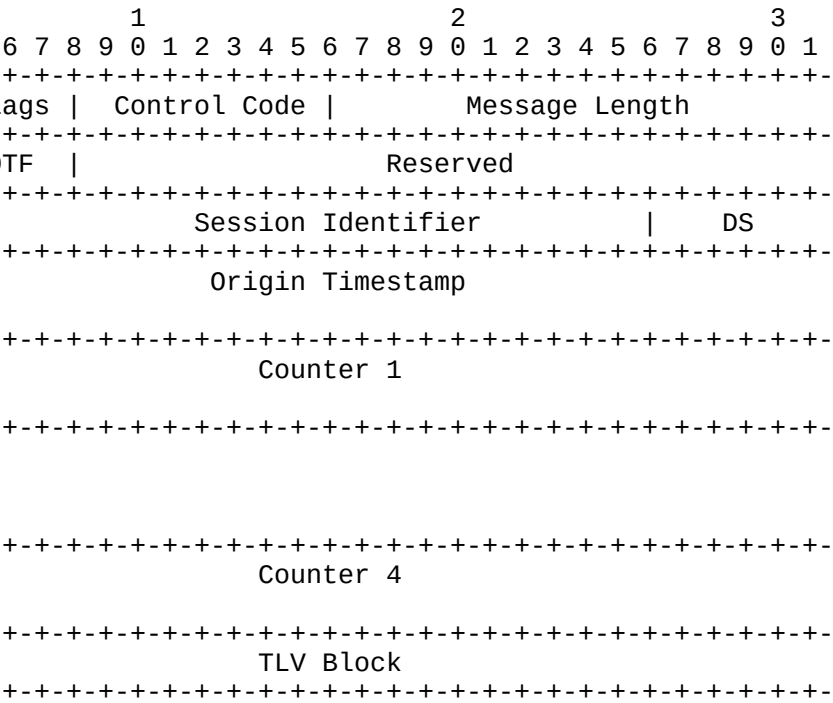


Figure 5: Loss Measurement Message Payload Format

The meanings of the fields are summarized in the following table, see

[RFC6374] for details.

Field	Meaning
Version	Protocol version
Flags	Message control flags
Control Code	Code identifying the query or response type
Message Length	Total length of this message in bytes
Data Format Flags (DFlags)	Flags specifying the format of message data
Origin Timestamp Format (OTF)	Format of the Origin Timestamp field
Reserved	Reserved for future specification
Session Identifier	Set arbitrarily by the querier
Differentiated Services (DS) Field	Differentiated Services Code Point (DSCP) being measured
Origin Timestamp	64-bit field for query message transmission timestamp
Counter 1-4	64-bit fields for LM counter values
TLV Block	Optional block of Type-Length-Value fields

6. Performance Measurement for P2MP SR Policies

The procedures for delay and loss measurement described in this document for Point-to-Point (P2P) SR-MPLS Policies are also equally applicable to the Point-to-Multipoint (P2MP) SR Policies.

The responder node may add the "Source Address" TLV (Type 130) [RFC6374] in the probe response message. This TLV allows the querier node to identify the responder node for the SR Policy.

7. SR Link Extended TE Metrics Advertisements

The extended TE metrics for SR link delay and loss computed using the performance measurement procedures reviewed in this document can be

advertised in the routing domain as follows:

- For OSPF, ISIS, and BGP-LS, protocol extensions defined in [RFC7471], [RFC7810] [I-D.lsr-isis-rfc7810bis], and [I-D.idr-te-pm-bgp] are used, respectively for advertising the extended TE link metrics in the network.

- The extended TE link delay metrics advertised are minimum-delay, maximum-delay, average-delay, and delay-variance for one-way.

- The delay-variance metric is computed as specified in Section 4.2 of [RFC5481].

- The one-way delay metrics can be computed using two-way measurement by dividing the measured delay values by 2.

- The extended TE link loss metric advertised is one-way percentage packet loss.

8. Security Considerations

This document reviews the procedures for performance delay and loss measurement for SR-MPLS networks, for both links and end-to-end SR Policies using the mechanisms defined in [RFC6374]. This document does not introduce any additional security considerations other than those covered in [RFC6374], [RFC7471], [RFC7810], and [RFC7876].

9. IANA Considerations

This document does not require any IANA actions.

10. References

10.1. Normative References

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS networks', RFC 6374, September 2011.

[RFC7876] Bryant, S., Sivabalan, S., and Soni, S., "UDP Return Path for Packet Loss and Delay Measurement for MPLS Networks", RFC 7876, July 2016.

10.2. Informative References

[IEEE1588] IEEE, "1588-2008 IEEE Standard for a Precision Clock

Synchronization Protocol for Networked Measurement and Control Systems", March 2008.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, March 2009.

[RFC7679] Almes, G., et al., "A One-Way Delay Metric for IP Performance Metrics (IPPM)', RFC 7679, January 2016.

[RFC7471] Giacalone, S., et al., "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, March 2015.

[RFC7810] Previdi, S., et al., "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 7810, May 2016.

[I-D.lsr-isis-rfc7810bis] Ginsberg, L., et al., "IS-IS Traffic Engineering (TE) Metric Extensions", draft-ietf-lsr-isis-rfc7810bis, work in progress.

[I-D.idr-te-pm-bgp] Ginsberg, L. Ed., et al., "BGP-LS Advertisement of IGP Traffic Engineering Performance Metric Extensions", draft-ietf-idr-te-pm-bgp, work in progress.

[I-D.spring-segment-routing-policy] Filsfils, C., et al., "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy, work in progress.

[I-D.spring-sr-p2mp-policy] Voyer, D. Ed., et al., "SR Replication Policy for P2MP Service Delivery", draft-voyer-spring-sr-p2mp-policy, work in progress.

[I-D.spring-mpls-path-segment] Cheng, W., et al., "Path Segment in MPLS Based Segment Routing Network", draft-cheng-spring-mpls-path-segment, work in progress.

[I-D.pce-sr-path-segment] Li, C., et al., "Path Computation Element Communication Protocol (PCEP) Extension for Path Identification in Segment Routing (SR)", draft-li-pce-sr-path-segment, work in progress.

Acknowledgments

To be added.

Contributors

Sagar Soni
Cisco Systems, Inc.
Email: sagsoni@cisco.com

Patrick Khordoc
Cisco Systems, Inc.
Email: pkhordoc@cisco.com

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

Daniel Bernier
Bell Canada
Email: daniel.bernier@bell.ca

Authors' Addresses

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

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

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

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

Pier Luigi Ventre
CNIT
Italy
Email: pierluigi.ventre@cnit.it

Mach(Guoyi) Chen
Huawei
Email: mach.chen@huawei.com

SPRING Working Group
Internet-Draft
Intended Status: Standards Track
Expires: December 11, 2018

R. Gandhi, Ed.
C. Filsfils
S. Soni
Cisco Systems, Inc.
D. Voyer
Bell Canada
S. Salsano
Universita di Roma "Tor Vergata"
P. L. Ventre
CNIT
June 9, 2018

UDP Path for In-band Performance Measurement for Segment Routing Networks

draft-gandhi-spring-udp-pm-01

Abstract

Segment Routing (SR) is applicable to both Multiprotocol Label Switching (SR-MPLS) and IPv6 (SRv6) data planes. This document specifies a procedure for using UDP path for sending and processing in-band probe query and response messages for Performance Measurement (PM). The procedure uses the RFC 6374 defined mechanisms for Delay and Loss performance measurement. The procedure specified is applicable to IPv4, IPv6, SR-MPLS, and SRv6 data planes for both links and end-to-end measurement for SR Policies. This document also defines mechanisms for handling Equal Cost Multipaths (ECMPs) for SR Policies. In addition, this document defines new Return Path Segment List TLV for two-way performance measurement.

Status of This Memo

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

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

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

Table of Contents

1. Introduction

Segment Routing (SR) technology greatly simplifies network operations for Software Defined Networks (SDNs). SR is applicable to both Multiprotocol Label Switching (SR-MPLS) and IPv6 (SRv6) data planes. SR takes advantage of the Equal-Cost Multipaths (ECMPs) between source, transit and destination nodes. SR Policies as defined in [I-D.spring-segment-routing-policy] are used to steer traffic through a specific, user-defined path using a stack of Segments. Built-in SR Performance Measurement (PM) is one of the essential requirements to provide Service Level Agreements (SLAs).

The One-Way Active Measurement Protocol (OWAMP) defined in [RFC4656] and Two-Way Active Measurement Protocol (TWAMP) defined in [RFC5357] provide capabilities for the measurement of various performance metrics in IP networks. These protocols rely on control channel signaling to establish a connection over an UDP path to bootstrap PM sessions, and they are not compatible with the mechanisms defined in [RFC6374]. These protocols lack support for IEEE 1588 timestamps [IEEE1588] and direct-mode LM, which are required in Segment Routing networks [RFC6374].

[RFC6374] specifies protocol mechanisms to enable the efficient and accurate measurement of performance metrics and can be used in SR networks with MPLS data plane [I-D.spring-sr-mpls-pm]. [RFC6374] addresses the limitations of the IP based performance measurement protocols. However, [RFC6374] requires data plane to support MPLS Generic Associated Channel Label (GAL) and Generic Associated Channel (G-Ach), which may not be supported on all nodes in the network.

[RFC7876] specifies the procedures to be used when sending and processing out-of-band performance measurement probe response messages over an UDP return path for RFC 6374 based probe queries. [RFC7876] can be used to send out-of-band PM probe responses in both SR-MPLS and SRv6 networks for one-way performance measurement.

For SR Policies, there is a need to measure the performance of all end-to-end forwarding paths due to presence of ECMPs between the source and transit nodes, between transit nodes and between transit and destination nodes. Existing PM protocols (e.g. OWAMP, TWAMP, RFC

6374, etc.) do not define handling for ECMP forwarding paths in SR networks.

For two-way measurements for SR Policies, there is a need to specify a return path in the form of a Segment List in PM probe query messages without requiring any SR Policy state on the destination node. Exiting protocols do not have such mechanisms to specify return path in the PM probe query messages.

This document specifies a procedure for using UDP path for sending and processing in-band probe query and response messages for Performance Measurement that does not require to bootstrap PM sessions. The procedure uses RFC 6374 defined mechanisms for Delay and Loss PM and unless otherwise specified, the procedures from RFC 6374 are not modified. The procedure specified is applicable to IPv4, IPv6, SR-MPLS and SRv6 data planes. The procedure does not require to bootstrap PM sessions and can be used for both SR links and end-to-end measurement for SR Policies. This document also defines mechanisms for handling Equal Cost Multipaths (ECMPs) for SR Policies. In addition, this document defines Return Path Segment List TLV for two-way performance measurement.

2. Conventions Used in This Document

2.1. 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 [RFC2119] [RFC8174].

2.2. Abbreviations

ACH: Associated Channel Header.

BSID: Binding Segment ID.

DFLag: Data Format Flag.

DM: Delay Measurement.

G-ACh: Generic Associated Channel (G-ACh).

GAL: Generic Associated Channel (G-ACh) Label.

LM: Loss Measurement.

MPLS: Multiprotocol Label Switching.

PM: Performance Measurement.

PTP: Precision Time Protocol.

RPSL: Return Path Segment List.

SID: Segment ID.

SL: Segment List.

SR: Segment Routing.

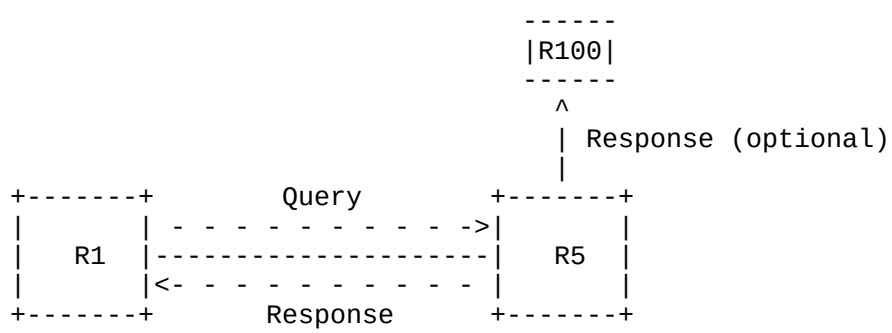
SR-MPLS: Segment Routing with MPLS data plane.

SRv6: Segment Routing with IPv6 data plane.

URO: UDP Return Object.

2.3. Reference Topology

In the reference topology, the querier node R1 initiates a probe query for performance measurement and the responder node R5 sends a probe response for the query message received. The probe response may be sent to the querier node R1 or to a controller node R100. The nodes R1 and R5 may be directly connected via a link enabled with Segment Routing or there exists an SR Policy [I-D.spring-segment-routing-policy] on node R1 with destination to node R5.



Reference Topology

Both Delay and Loss performance measurement is performed in-band for the traffic traversing between node R1 and node R5. One-way delay

and two-way delay measurements are defined in Section 2.4 of [RFC6374]. Transmit and Receive packet loss measurements are defined in Section 2.2 of [RFC6374]. One-way loss measurement provides receive packet loss whereas two-way loss measurement provides both transmit and receive packet loss.

3. Probe Messages

3.1. Probe Query Message

In this document, UDP path is defined for sending and processing PM probe query messages for Delay and Loss measurements for SR links and end-to-end SR Policies as described in the following Sections. As well-known UDP port is used for identifying PM probe packets, bootstrapping of the PM session [RFC5357] is not required.

3.1.1. Delay Measurement Probe Query Message

The message content for Delay Measurement probe query message using UDP header [RFC768] is shown in Figure 1. As shown, the DM probe query message is sent with Destination UDP port number TBA1 defined in this document. The Source UDP port may optionally be set to TBA1 for two-way delay measurement. The DM probe query message contains the payload for delay measurement defined in Section 3.2 of [RFC6374].

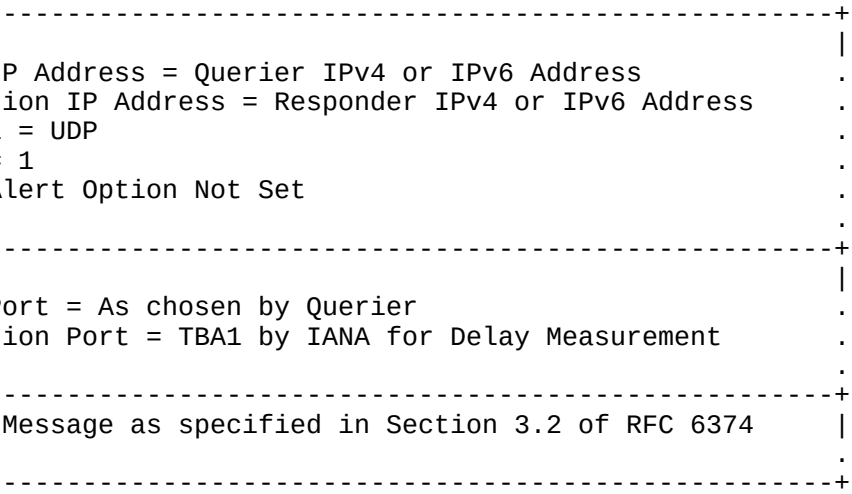


Figure 1: DM Probe Query Message

3.1.2. Loss Measurement Probe Query Message

The message content for Loss measurement probe query message using UDP header [RFC768] is shown in Figure 2. As shown, the LM probe query message is sent with Destination UDP port number TBA2 defined in this document. The Source UDP port may optionally be set to TBA2 for two-way loss measurement. The LM probe query message contains the payload for loss measurement defined in Section 3.1 of [RFC6374].

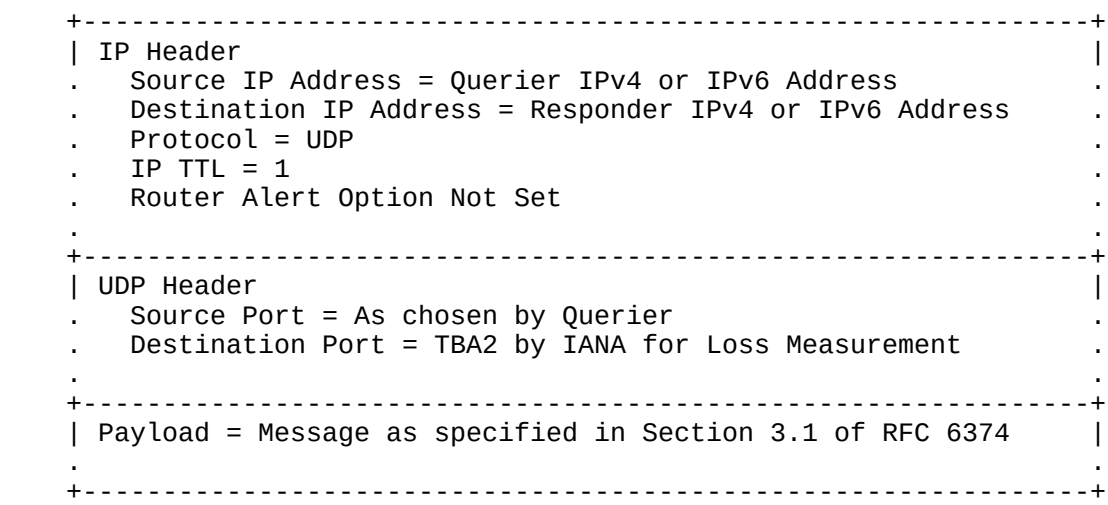


Figure 2: LM Probe Query Message

3.1.2.1. Loss Measurement Flags

An LM message carries Data Format Flags (DFlags) as defined in [RFC6374]. New Flag is defined in this document for Color (C) in the DFlags field as follows.

Data Format Flags

The Flag C indicates the Color of the counters in the LM probe message [RFC6374] when using Alternate-Marking method defined in [RFC8321].

3.1.3. In-band Probe Query for SR Links

The probe query message defined in Figure 1 is sent in-band for Delay measurement and defined in Figure 2 is used for Loss measurement for SR links.

3.1.4. In-band Probe Query for End-to-end Measurement of SR Policy

3.1.4.1. In-band Probe Query Message for SR-MPLS Policy

The message content for in-band probe query message using UDP header for end-to-end performance measurement of SR-MPLS Policy is shown in Figure 3.

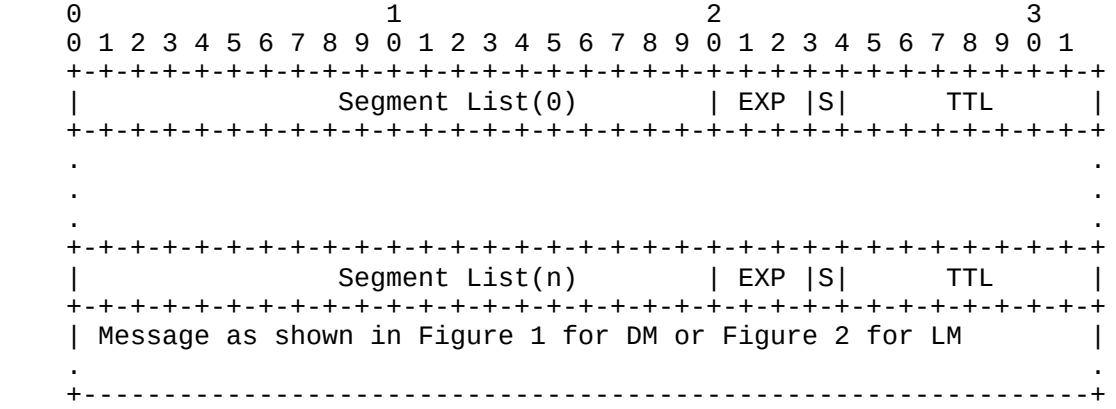


Figure 3: In-band Probe Query Message for SR-MPLS Policy

The Segment List (SL) can be empty to indicate Implicit NULL label case.

3.1.4.2. In-band Probe Query Message for SRv6 Policy

The in-band probe query messages using UDP header for end-to-end performance measurement of an SRv6 Policy is sent using SRv6 Segment Routing Header (SRH) and Segment List as defined in [I-D.6man-segment-routing-header] and is shown in Figure 4.

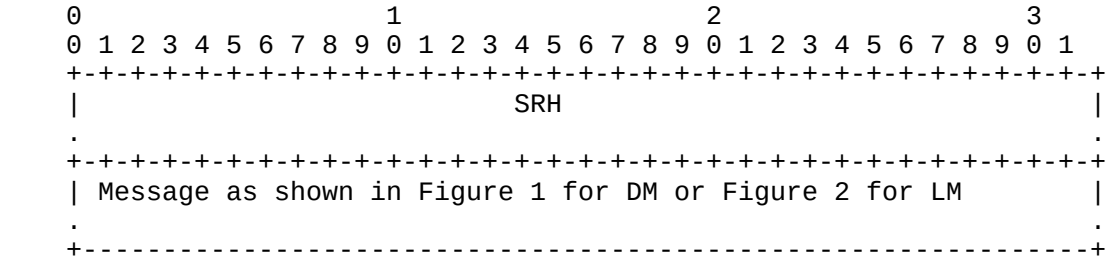


Figure 4: In-band Probe Query Message for SRv6 Policy

3.2. Probe Response Message

When the received probe query message does not contain any UDP Return Object (URO) TLV [RFC7876], the probe response message uses the IP/UDP information from the probe query message. The content of the probe response message is shown in Figure 5.

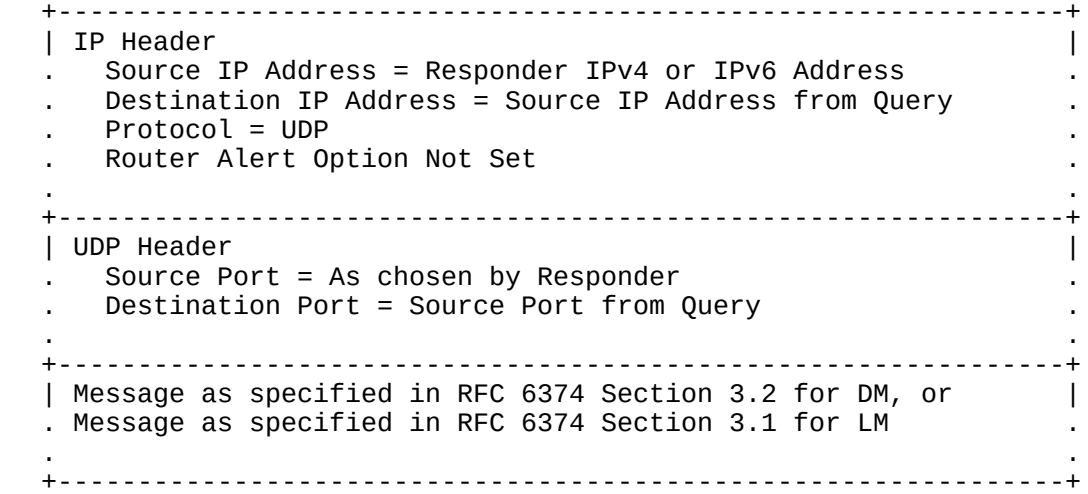


Figure 5: Probe Response Message

When the received probe query message contains UDP Return Object (URO) TLV [RFC7876], the probe response message the message uses the IP/UDP information from the URO in the probe query message. The content of the probe response message is shown in Figure 6.

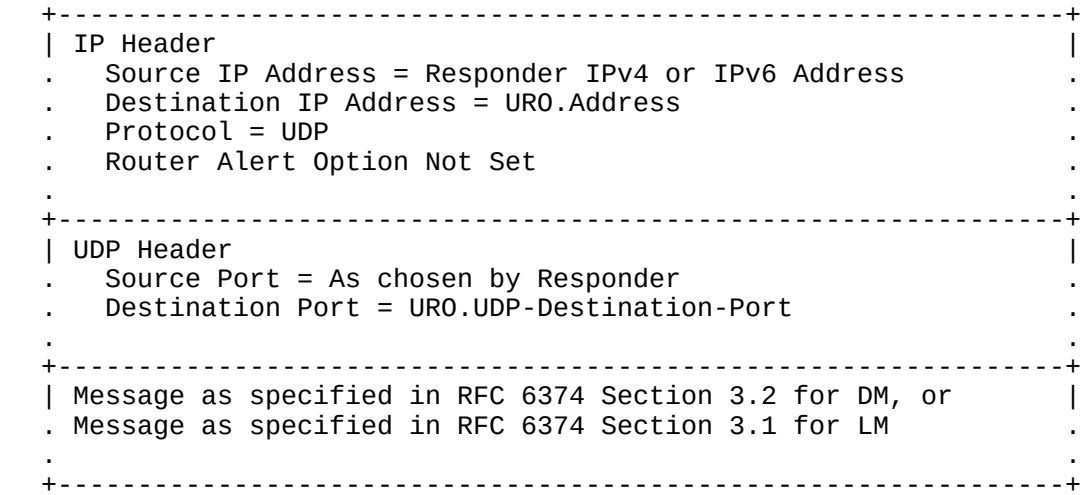


Figure 6: Probe Response Message Using URO from Probe Query Message

3.2.1. One-way Measurement for SR Link and end-to-end SR Policy

For one-way performance measurement, the probe response message as defined in Figure 5 or Figure 6 is sent out-of-band for both SR links and SR Policies.

The PM querier node can receive probe response message back by properly setting its own IP address as Source Address of the header or by adding URO TLV in the probe query message and setting its own IP address in the IP Address in the URO TLV (Type=131) [RFC7876].

3.2.1.1. Probe Response Message to Controller

As shown in Reference Topology, if the querier node requires the probe response message to be sent to the controller R100, it adds URO TLV in the probe query message and sets the IP address of R100 in the IP Address field and UDP port TBA1 for DM and TBA2 for LM in the UDP-Destination-Port field of the URO TLV (Type=131) [RFC7876].

3.2.2. Two-way Measurement for SR Links

For two-way performance measurement, when using a bidirectional channel, the probe response message as defined in Figure 5 or Figure 6 is sent back in-band to the querier node for SR links. In this case, the "control code" in the probe query message is set to "in-band response requested" [RFC6374].

3.2.3. Two-way End-to-end Measurement of SR Policy

For two-way performance measurement, when using a bidirectional channel, the probe response message is sent back in-band to the querier node for end-to-end measurement of SR Policies. In this case, the "control code" in the probe query message is set to "in-band response requested" [RFC6374].

3.2.3.1. Return Path Segment List TLV

For two-way performance measurement, the responder node needs to send the probe response message in-band on a specific reverse SR path. This way the destination node does not require any additional SR Policy state. The querier node can request in the probe query message to the responder node to send a response back on a given reverse path (typically co-routed path for two-way measurement).

[RFC6374] defines DM and LM probe query messages that can include one or more optional TLVs. New TLV Types are defined in this document

for Return Path Segment List (RPSL) to carry reverse SR path for probe response messages. The format of the RPSL TLV is shown in Figure 7:

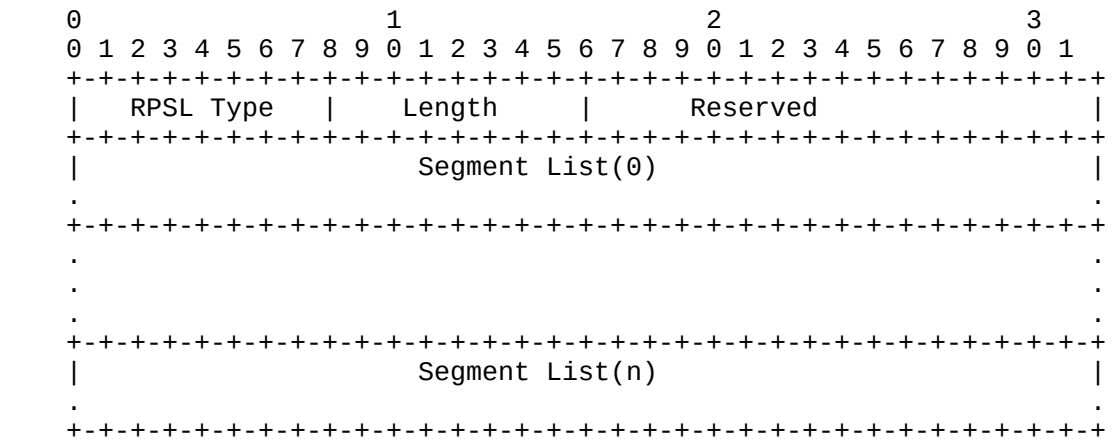


Figure 7: Return Path Segment List TLV

All Segments in Segment List can be one of following Types:

- RPSL Type (value TBA3) carrying SR-MPLS Labels
- RPSL Type (value TBA4) carrying SRv6 Segments
- RPSL Type (value TBA5) carrying SR-MPLS Binding SID [I-D.pce-binding-label-sid] of the Reverse SR Policy
- RPSL Type (value TBA6) carrying SRv6 Binding SID of the Reverse SR Policy

The Segment List(0) can be used by the responder node to compute the next-hop IP address and outgoing interface to send the probe response messages.

The RPSL TLV is optional. The PM querier node MUST only insert one RPSL TLV in the probe query message and the responder node MUST only process the first RPSL TLV in the probe query message and ignore other RPSL TLVs if present. The responder node MUST send probe response message back on the reverse path specified in the RPSL TLV and MUST NOT add RPSL TLV in the probe response message.

3.2.3.2. In-band Probe Response Message for SR-MPLS Policy

The message content for sending probe response message in-band using

UDP header for two-way end-to-end performance measurement of an SR-MPLS Policy is shown in Figure 8. The SR-MPLS label stack in the packet header is built using the Segment List received in the RPSL TLV in the probe query message.

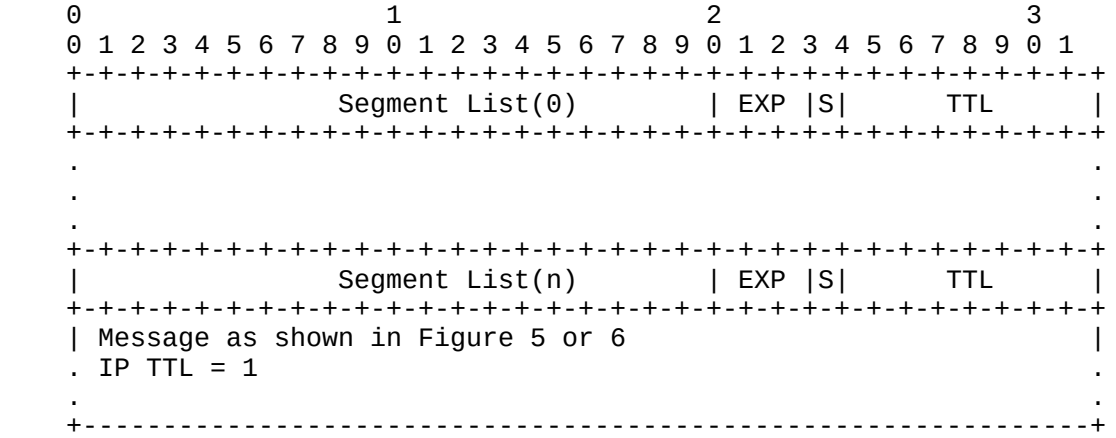


Figure 8: In-band Probe Response Message for SR-MPLS Policy

3.2.3.3. In-band Probe Response Message for SRv6 Policy

The message content for sending probe response message in-band using UDP header for two-way end-to-end performance measurement of an SRv6 Policy is shown in Figure 9. For SRv6 Policy, the SRv6 SID stack in the probe response message SRH is built using the SRv6 Segment List received in the RPSL TLV in the probe query message.

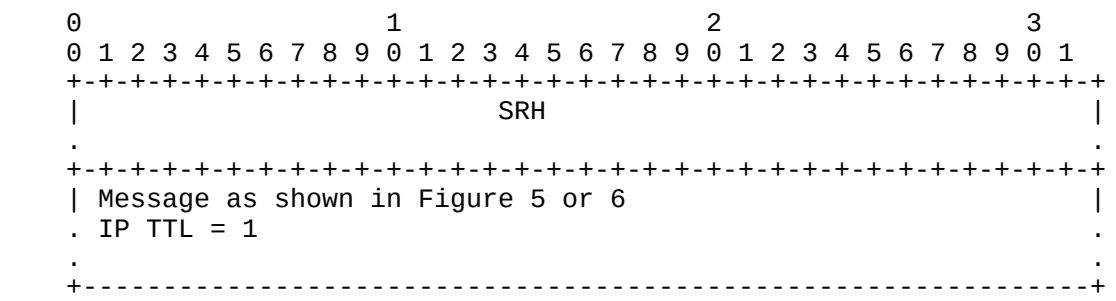


Figure 9: In-band Probe Response Message for SRv6 Policy

3.3. ECMP Support

An SR Policy can have a number of end-to-end forwarding paths due to

presence of Equal Cost Multipaths (ECMPs) between the source and transit nodes, between transit nodes and between transit and destination nodes. The PM probe messages can be sent to traverse different ECMP forwarding paths and measure performance of all end-to-end forwarding paths of an SR Policy.

Forwarding planes have various hashing functions available to forward packets on specific ECMP paths. Following mechanisms can be used in PM probe messages to take advantage of the hashing function in forwarding plane to influence the ECMP path taken by them.

- For IPv4 and SR-MPLS, the mechanisms described in [RFC8029] for handling ECMPs are also applicable to the performance measurement. For IPv4 and SR-MPLS, in IP/UDP header of the PM probe messages, different Destination Addresses in the range of 127/8 [RFC8029] or different Source Addresses or different Source UDP ports, etc. can be used.

- For SR-MPLS, entropy label [RFC6790] in the PM probe messages can be used.

- For IPv6, as specified in [RFC6437], 3-tuple of Flow Label, Source Address and Destination Address fields in the IPv6 header of the PM probe messages can be used.

- For SRv6, Flow Label in SRH [I-D.6man-segment-routing-header] of the PM probe messages can be used.

3.4. Sequence Number TLV

The message formats for DM and LM [RFC6374] do not contain sequence number for probe query packets. Sequence numbers can be useful when some probe query messages are lost or they arrive out of order.

[RFC6374] defines DM and LM probe query and response messages that can include one or more optional TLVs. New TLV Type (value TBA7) is defined in this document to carry sequence number for probe query and response messages. The format of the Sequence Number TLV is shown in Figure 10:

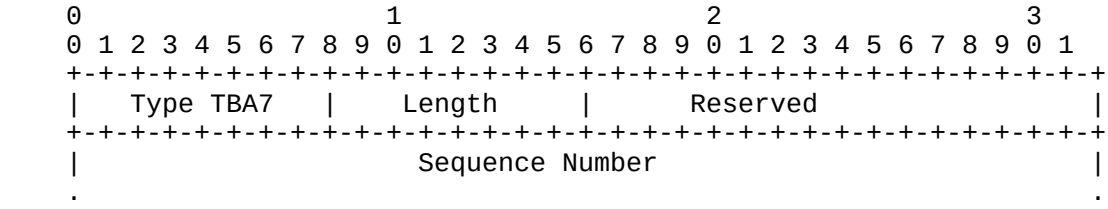


Figure 10: Sequence Number TLV

The Sequence Number TLV is optional. The PM querier node SHOULD only insert one Sequence Number TLV in the probe query message and the responder node in the probe response message SHOULD return the first Sequence Number TLV from the probe query messages and ignore other Sequence Number TLVs if present.

4. Security Considerations

The performance measurement is intended for deployment in well-managed private and service provider networks. The security considerations described in Section 8 of [RFC6374] are applicable to this specification, and particular attention should be paid to the last two paragraphs. Cryptographic measures may be enhanced by the correct configuration of access-control lists and firewalls.

5. IANA Considerations

IANA is requested to allocate following UDP ports for performance measurements:

- UDP Port TBA1: Delay Performance Measurement
- UDP Port TBA2: Loss Performance Measurement

IANA is also requested to allocate values for the following Return Path Segment List TLV Types for RFC 6374 to be carried in PM probe query messages:

- Type TBA3: SR-MPLS Segment List of the Reverse SR Policy
- Type TBA4: SRv6 Segment List of the Reverse SR Policy
- Type TBA5: SR-MPLS Binding SID of the Reverse SR Policy
- Type TBA6: SRv6 Binding SID of the Reverse SR Policy

IANA is also requested to allocate a value for the following Sequence Number TLV Type for RFC 6374 to be carried in PM probe query and response messages:

- Type TBA7: Sequence Number TLV

6. References

6.1. Normative References

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

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

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS networks', RFC 6374, September 2011.

[RFC7876] Bryant, S., Sivabalan, S., and Soni, S., "UDP Return Path for Packet Loss and Delay Measurement for MPLS Networks", RFC 7876, July 2016.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017.

6.2. Informative References

[IEEE1588] IEEE, "1588-2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", March 2008.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protoco (OWAMP)", RFC 4656, September 2006.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, November 2011.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

[RFC8029] Kompella, K., Swallow, G., Pignataro, C., Kumar, N., Aldrin, S. and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, March 2017.

[RFC8321] Fioccola, G. Ed., "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, January

2018.

[I-D.spring-segment-routing-policy] Filsfils, C., et al., "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy, work in progress.

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

[I-D.spring-sr-mpls-pm] Filsfils, C., Gandhi, R. Ed., et al. "Performance Measurement in Segment Routing Networks with MPLS Data Plane", draft-gandhi-spring-sr-mpls-pm, work in progress.

[I-D.pce-binding-label-sid] Filsfils, C., et al., "Carrying Binding Label Segment-ID in PCE-based Networks", draft-sivabalan-pce-binding-label-sid, work in progress.

Acknowledgments

The authors would like to thank Faisal Iqbal, Nagendra Kumar and Carlos Pignataro for the discussion on SRv6 Performance Measurement.

Contributors

Patrick Khordoc
Cisco Systems, Inc.
Email: pkhordoc@cisco.com

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

Daniel Bernier
Bell Canada
Email: daniel.bernier@bell.ca

Dirk Steinberg
Steinberg Consulting
Germany
Email: dws@dirksteinberg.de

Authors' Addresses

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

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

Sagar Soni
Cisco Systems, Inc.
Email: sagsoni@cisco.com

Daniel Voyer

Bell Canada
Email: daniel.voyer@bell.ca

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy
Email: stefano.salsano@uniroma2.it

Pier Luigi Ventre
CNIT
Italy
Email: pierluigi.ventre@cnit.it

SPRING Working Group
Internet-Draft
Intended Status: Standards Track
Expires: March 18, 2019

R. Gandhi, Ed.
C. Filsfils
Cisco Systems, Inc.
D. Voyer
Bell Canada
S. Salsano
Universita di Roma "Tor Vergata"
P. L. Ventre
CNIT
M. Chen
Huawei

September 14, 2018

UDP Path for In-band Performance Measurement for Segment Routing Networks

draft-gandhi-spring-udp-pm-02

Abstract

Segment Routing (SR) is applicable to both Multiprotocol Label Switching (SR-MPLS) and IPv6 (SRv6) data planes. This document specifies procedures for using UDP path for sending and processing in-band probe query and response messages for Performance Measurement. The procedure uses the RFC 6374 defined mechanisms for Delay and Loss performance measurement. The procedure specified is applicable to SR-MPLS and SRv6 data planes for both links and end-to-end measurement for SR Policies. This document also defines mechanisms for handling Equal Cost Multipaths (ECMPs) for SR Policies. In addition, this document defines Return Path Segment List TLV for two-way performance measurement and Block Number TLV for loss measurement.

Status of This Memo

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

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

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

Table of Contents

1. Introduction

Segment Routing (SR) technology greatly simplifies network operations for Software Defined Networks (SDNs). SR is applicable to both Multiprotocol Label Switching (SR-MPLS) and IPv6 (SRv6) data planes. SR takes advantage of the Equal-Cost Multipaths (ECMPs) between source, transit and destination nodes. SR Policies as defined in [I-D.spring-segment-routing-policy] are used to steer traffic through a specific, user-defined path using a stack of Segments. Built-in SR Performance Measurement (PM) is one of the essential requirements to provide Service Level Agreements (SLAs).

The One-Way Active Measurement Protocol (OWAMP) defined in [RFC4656] and Two-Way Active Measurement Protocol (TWAMP) defined in [RFC5357] provide capabilities for the measurement of various performance metrics in IP networks. These protocols rely on control channel signaling to establish a test channel over an UDP path. These protocols lack support for IEEE 1588 timestamp [IEEE1588] format and direct-mode Loss Measurement (LM), which are required in SR networks [RFC6374]. The Simple Two-way Active Measurement Protocol (STAMP) [I-D.ippm-stamp] alleviates the control channel signaling by using configuration data model to provision test channels. In addition, the STAMP supports IEEE 1588 timestamp format for Delay Measurement (DM). The TWAMP Light from broadband forum [BBF.TR-390] provides simplified mechanisms for active performance measurement in Customer Edge IP networks.

[RFC6374] specifies protocol mechanisms to enable the efficient and accurate measurement of performance metrics and can be used in SR networks with MPLS data plane [I-D.spring-sr-mpls-pm]. [RFC6374] addresses the limitations of the IP based performance measurement protocols as specified in Section 1 of [RFC6374]. The [RFC6374] requires data plane to support MPLS Generic Associated Channel Label (GAL) and Generic Associated Channel (G-Ach), which may not be supported on all nodes in the network.

[RFC7876] specifies the procedures to be used when sending and processing out-of-band performance measurement probe response messages over an UDP return path for RFC 6374 based probe queries.

[RFC7876] can be used to send out-of-band PM probe responses in both SR-MPLS and SRv6 networks for one-way performance measurement.

For SR Policies, there are ECMPs between the source and transit nodes, between transit nodes and between transit and destination nodes. Existing PM protocols (e.g. RFC 6374) do not define handling for ECMP forwarding paths in SR networks.

For two-way measurements for SR Policies, there is a need to specify a return path in the form of a Segment List in PM probe query messages without requiring any SR Policy state on the destination node. Exiting protocols do not have such mechanisms to specify return path in the PM probe query messages.

This document specifies a procedure for using UDP path for sending and processing in-band probe query and response messages for Performance Measurement that does not require to bootstrap PM sessions. The procedure uses RFC 6374 defined mechanisms for Delay and Loss PM and unless otherwise specified, the procedures from RFC 6374 are not modified. The procedure specified is applicable to both SR-MPLS and SRv6 data planes. The procedure does not require to bootstrap PM sessions and can be used for both SR links and end-to-end performance measurement for SR Policies. This document also defines mechanisms for handling Equal Cost Multipaths (ECMPs) for SR Policies. In addition, this document defines Return Path Segment List (RPSL) TLV for two-way performance measurement and Block Number TLV for loss measurement.

2. Conventions Used in This Document

2.1. 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 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Abbreviations

ACH: Associated Channel Header.

BSID: Binding Segment ID.

DFLag: Data Format Flag.

DM: Delay Measurement.

ECMP: Equal Cost Multi-Path.

G-ACh: Generic Associated Channel (G-ACh).

GAL: Generic Associated Channel (G-ACh) Label.

LM: Loss Measurement.

MPLS: Multiprotocol Label Switching.

NTP: Network Time Protocol.

OWAMP: One-Way Active Measurement Protocol.

PM: Performance Measurement.

PTP: Precision Time Protocol.

RPSL: Return Path Segment List.

SID: Segment ID.

SL: Segment List.

SR: Segment Routing.

SR-MPLS: Segment Routing with MPLS data plane.

SRv6: Segment Routing with IPv6 data plane.

STAMP: Simple Two-way Active Measurement Protocol.

TC: Traffic Class.

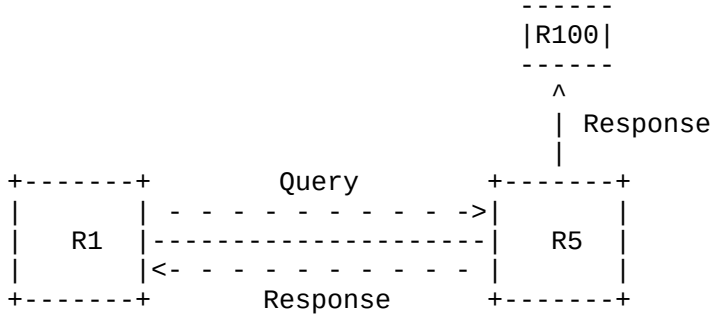
TWAMP: Two-Way Active Measurement Protocol.

URO: UDP Return Object.

2.3. Reference Topology

In the reference topology, the querier node R1 initiates a probe query for performance measurement and the responder node R5 sends a probe response for the query message received. The probe response may be sent to the querier node R1 or to a controller node R100. The nodes R1 and R5 may be directly connected via a link enabled with Segment Routing or there exists a Point-to-Point (P2P) SR Policy [I-D.spring-segment-routing-policy] on node R1 with destination to

node R5. In case of Point-to-Multipoint (P2MP), SR Policy originating from source node R1 may terminate on multiple destination leaf nodes [I-D.spring-sr-p2mp-policy].



Reference Topology

Both Delay and Loss performance measurement is performed in-band for the traffic traversing between node R1 and node R5. One-way delay and two-way delay measurements are defined in Section 2.4 of [RFC6374]. Transmit and Receive packet loss measurements are defined in Section 2.2 and Section 2.6 of [RFC6374]. One-way loss measurement provides receive packet loss whereas two-way loss measurement provides both transmit and receive packet loss.

3. Probe Messages

3.1. Probe Query Message

In this document, UDP path is defined for sending and processing PM probe query messages for Delay and Loss measurements for SR links and end-to-end SR Policies as described in the following Sections. As well-known UDP port is used for identifying PM probe packets, bootstrapping of the PM session [RFC5357] is not required. The TTL / Hop Limit field of the IP header MUST be set to 1.

3.1.1. Delay Measurement Probe Query Message

The message content for Delay Measurement probe query message using UDP header [RFC768] is shown in Figure 1. As shown, the DM probe query message is sent with Destination UDP port number TBA1 defined in this document. The Source UDP port may optionally be set to TBA1 for two-way delay measurement. The DM probe query message contains the payload for delay measurement defined in Section 3.2 of [RFC6374].

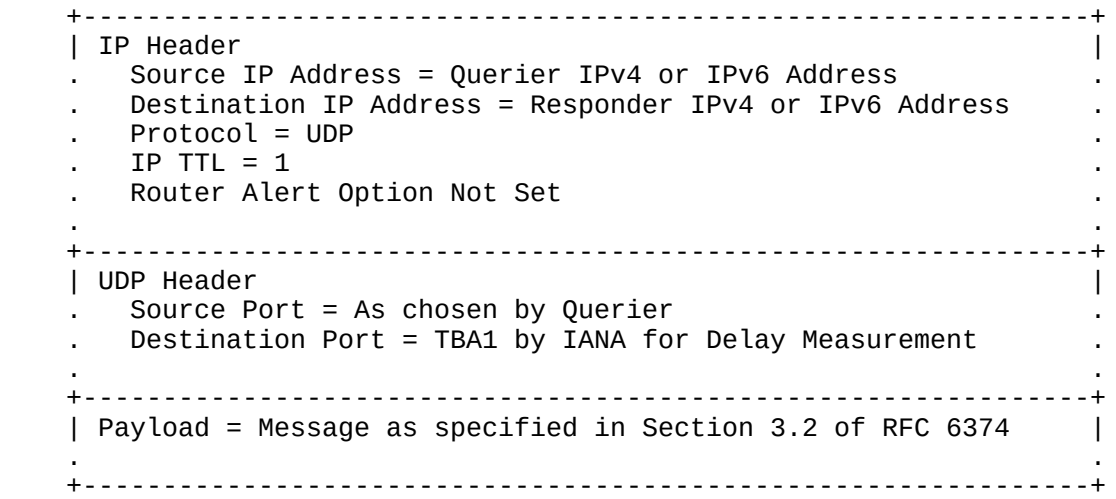


Figure 1: DM Probe Query Message

3.1.2. Loss Measurement Probe Query Message

The message content for Loss measurement probe query message using UDP header [RFC768] is shown in Figure 2. As shown, the LM probe query message is sent with Destination UDP port number TBA2 defined in this document. The Source UDP port may optionally be set to TBA2 for two-way loss measurement. The LM probe query message contains the payload for loss measurement defined in Section 3.1 of [RFC6374].

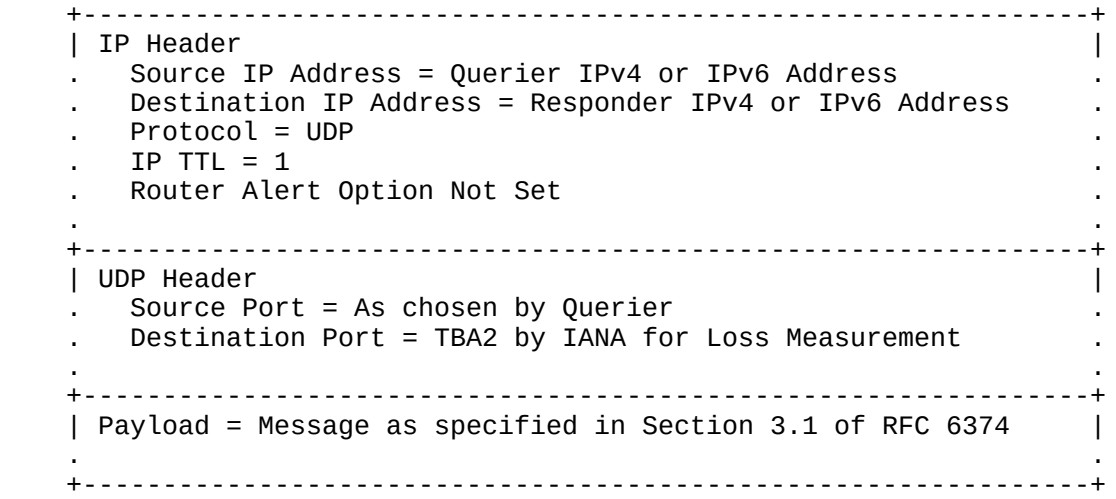


Figure 2: LM Probe Query Message

The path segment identifier [I-D.spring-mpls-path-segment] [I-D.pce-sr-path-segment] of the SR Policy is required for accounting received traffic on the egress node for loss measurement.

3.1.2.1. Block Number TLV

The Loss Measurement using Alternate-Marking method defined in [RFC8321] requires to identify the Block Number (color) of the traffic counters carried by the probe query and response messages. Probe query and response messages specified in [RFC6374] for Loss Measurement do not define any means to carry the Block Number.

[RFC6374] defines probe query and response messages that can include one or more optional TLVs. New TLV Type (value TBA8) is defined in this document to carry Block Number (32-bit) for the traffic counters in the probe query and response messages for loss measurement. The format of the Block Number TLV is shown in Figure 11:

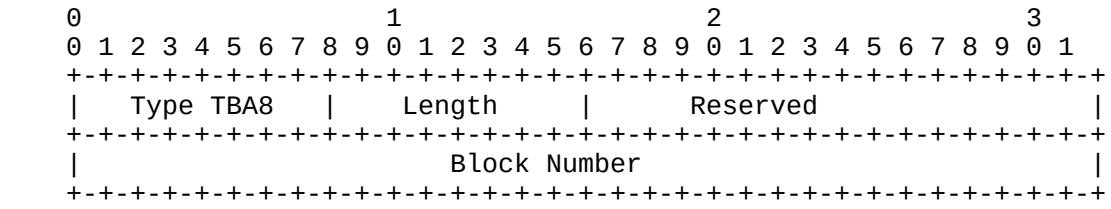


Figure 11: Block Number TLV

The Block Number TLV is optional. The PM querier node SHOULD only insert one Block Number TLV in the probe query message and the responder node in the probe response message SHOULD return the first Block Number TLV from the probe query messages and ignore other Block Number TLVs if present. In both probe query and response messages, the counters MUST belong to the same Block Number.

3.1.3. In-band Probe Query for SR Links

The probe query message as defined in Figure 1 is sent in-band for Delay measurement. The probe query message as defined in Figure 2 is sent in-band for Loss measurement.

3.1.4. In-band Probe Query for End-to-end Measurement of SR Policy

3.1.4.1. In-band Probe Query Message for SR-MPLS Policy

The message content for in-band probe query message using UDP header

for end-to-end performance measurement of SR-MPLS Policy is shown in Figure 3.

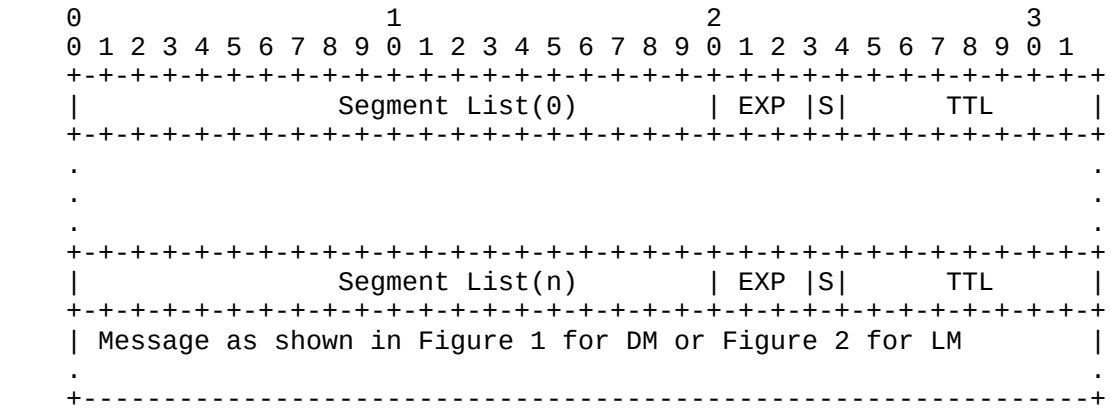


Figure 3: In-band Probe Query Message for SR-MPLS Policy

The Segment List (SL) can be empty to indicate Implicit NULL label case.

3.1.4.2. In-band Probe Query Message for SRv6 Policy

The in-band probe query messages using UDP header for end-to-end performance measurement of an SRv6 Policy is sent using SRv6 Segment Routing Header (SRH) and Segment List of the SRv6 Policy as defined in [I-D.6man-segment-routing-header] and is shown in Figure 4.

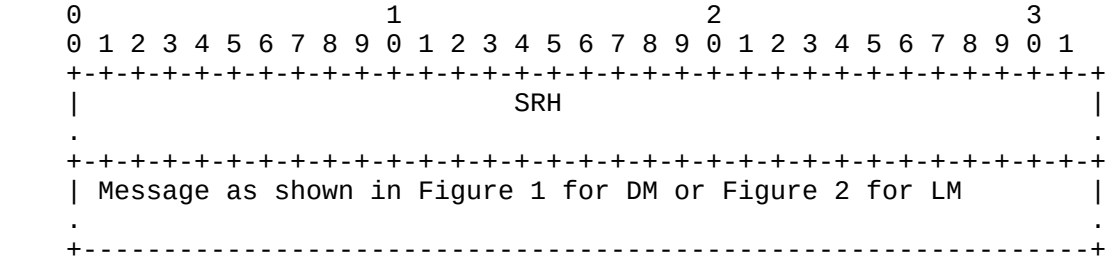


Figure 4: In-band Probe Query Message for SRv6 Policy

3.2. Probe Response Message

When the received probe query message does not contain any UDP Return Object (URO) TLV [RFC7876], the probe response message is sent using the IP/UDP information from the probe query message. The content of

the probe response message is shown in Figure 5.

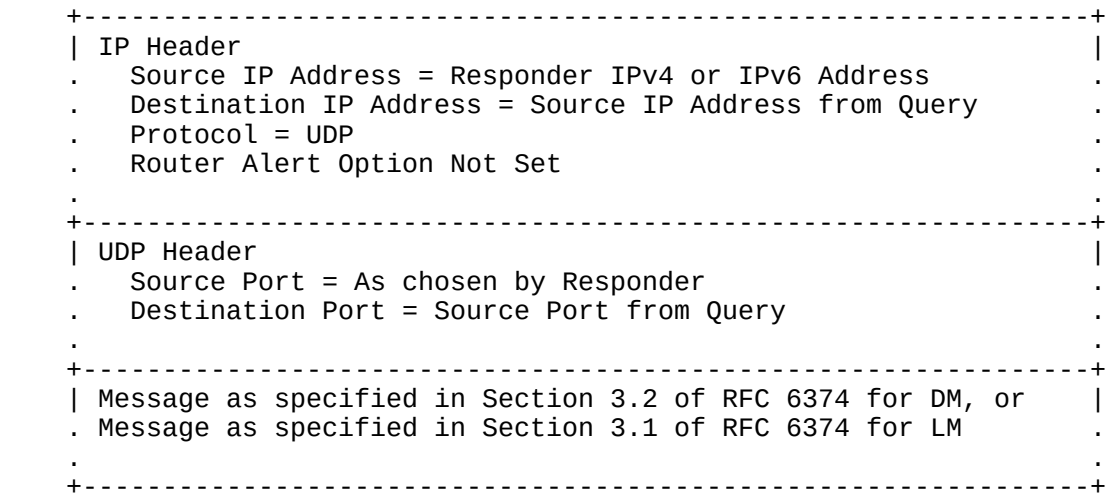


Figure 5: Probe Response Message

When the received probe query message contains UDP Return Object (URO) TLV [RFC7876], the probe response message the message uses the IP/UDP information from the URO in the probe query message. The content of the probe response message is shown in Figure 6.

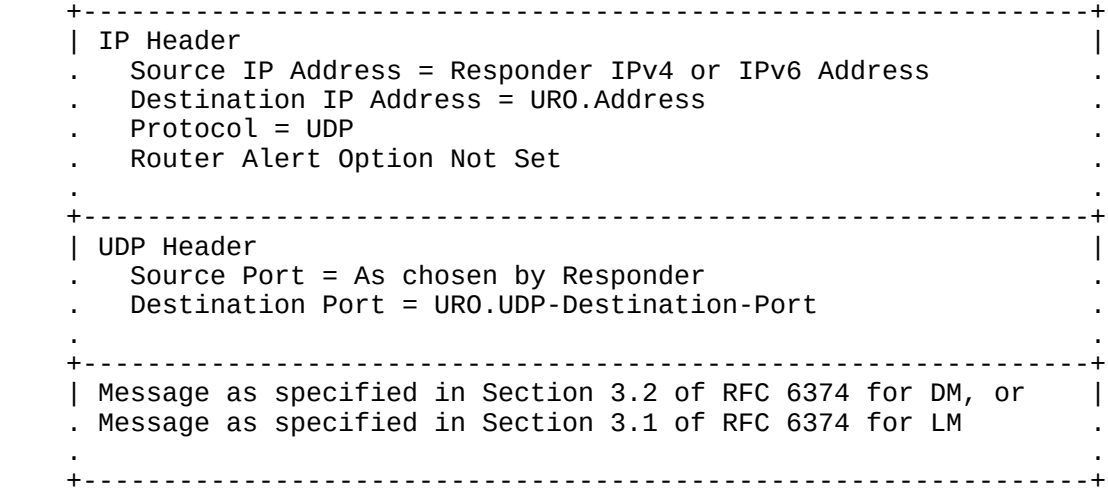


Figure 6: Probe Response Message Using URO from Probe Query Message

3.2.1. One-way Measurement for SR Link and end-to-end SR Policy

For one-way performance measurement, the probe response message as defined in Figure 5 or Figure 6 is sent out-of-band for both SR links and SR Policies.

The PM querier node can receive probe response message back by properly setting its own IP address as Source Address of the header or by adding URO TLV in the probe query message and setting its own IP address in the IP Address in the URO TLV (Type=131) [RFC7876]. In addition, the "control code" in the probe query message is set to "out-of-band response requested". The "Source Address" TLV (Type 130), and "Return Address" TLV (Type 1), if present in the probe query message, are not used to send probe response message.

3.2.1.1. Probe Response Message to Controller

As shown in the Reference Topology, if the querier node requires the probe response message to be sent to the controller R100, it adds URO TLV in the probe query message and sets the IP address of R100 in the IP Address field and UDP port TBA1 for DM and TBA2 for LM in the UDP-Destination-Port field of the URO TLV (Type=131) [RFC7876].

3.2.2. Two-way Measurement for SR Links

For two-way performance measurement, when using a bidirectional channel, the probe response message as defined in Figure 5 or Figure 6 is sent back in-band to the querier node for SR links. In this case, the "control code" in the probe query message is set to "in-band response requested" [RFC6374].

3.2.3. Two-way End-to-end Measurement of SR Policy

For two-way performance measurement, when using a bidirectional channel, the probe response message is sent back in-band to the querier node for end-to-end measurement of SR Policies. In this case, the "control code" in the probe query message is set to "in-band response requested" [RFC6374].

The path segment identifier [I-D.spring-mpls-path-segment] [I-D.pce-sr-path-segment] of the forward SR Policy can be used to find the reverse SR Policy to send the probe response message in the absence of RPSL TLV defined in the following Section.

3.2.3.1. Return Path Segment List TLV

For two-way performance measurement, the responder node needs to send the probe response message in-band on a specific reverse SR path. This way the destination node does not require any additional SR Policy state. The querier node can request in the probe query

message to the responder node to send a response back on a given reverse path (typically co-routed path for two-way measurement).

[RFC6374] defines DM and LM probe query messages that can include one or more optional TLVs. New TLV Types are defined in this document for Return Path Segment List (RPSL) to carry reverse SR path for probe response messages. The format of the RPSL TLV is shown in Figure 7:

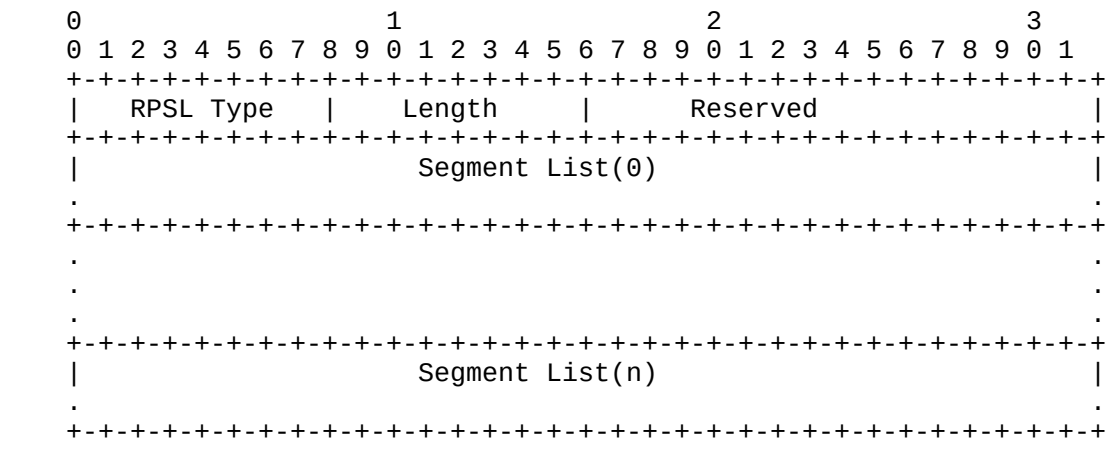


Figure 7: Return Path Segment List TLV

The RPSL can be one of following Types:

- RPSL Type (value TBA3): SR-MPLS Label Stack of the Reverse SR Policy
- RPSL Type (value TBA4): SRv6 Segment List of the Reverse SR Policy
- RPSL Type (value TBA5): SR-MPLS Binding SID [I-D.pce-binding-label-sid] of the Reverse SR Policy
- RPSL Type (value TBA6): SRv6 Binding SID [I-D.pce-binding-label-sid] of the Reverse SR Policy

The Segment List(0) can be used by the responder node to compute the next-hop IP address and outgoing interface to send the probe response messages.

The RPSL TLV is optional. The PM querier node MUST only insert one RPSL TLV in the probe query message and the responder node MUST only process the first RPSL TLV in the probe query message and ignore

other RPSL TLVs if present. The responder node MUST send probe response message back on the reverse path specified in the RPSL TLV and MUST NOT add RPSL TLV in the probe response message.

3.2.3.2. In-band Probe Response Message for SR-MPLS Policy

The message content for sending probe response message in-band using UDP header for two-way end-to-end performance measurement of an SR-MPLS Policy is shown in Figure 8. The SR-MPLS label stack in the packet header is built using the Segment List received in the RPSL TLV in the probe query message.

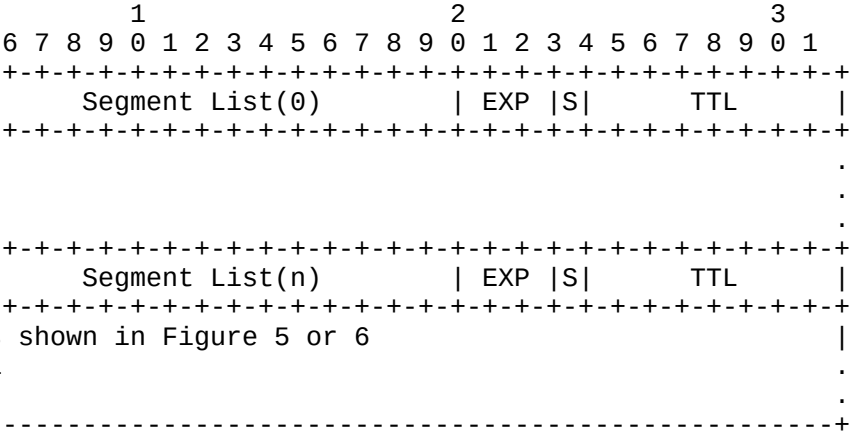


Figure 8: In-band Probe Response Message for SR-MPLS Policy

3.2.3.3. In-band Probe Response Message for SRv6 Policy

The message content for sending probe response message in-band using UDP header for two-way end-to-end performance measurement of an SRv6 Policy is shown in Figure 9. For SRv6 Policy, the SRv6 SID list in the SRH of the probe response message is built using the SRv6 Segment List received in the RPSL TLV in the probe query message.

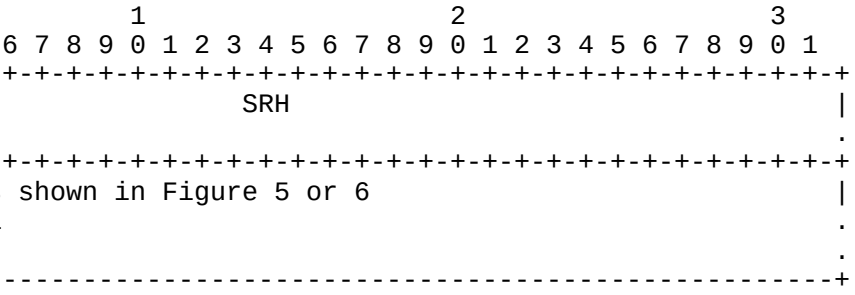


Figure 9: In-band Probe Response Message for SRv6 Policy

4. Performance Measurement for P2MP SR Policies

The procedures for delay and loss measurement described in this document for Point-to-Point (P2P) SR-MPLS Policies are also equally applicable to the Point-to-Multipoint (P2MP) SR Policies.

5. ECMP Support

An SR Policy can have ECMPs between the source and transit nodes, between transit nodes and between transit and destination nodes. The PM probe messages can be sent to traverse different ECMP paths to measure performance of an SR Policy.

Forwarding plane has various hashing functions available to forward packets on specific ECMP paths. Following mechanisms can be used in PM probe messages to take advantage of the hashing function in forwarding plane to influence the path taken by them.

- The mechanisms described in [RFC8029] [RFC5884] for handling ECMPs are also applicable to the performance measurement. In the IP/UDP header of the PM probe messages, Destination Addresses in 127/8 range for IPv4 or 0:0:0:0:0:FFFF:7F00/104 range for IPv6 can be used to exercise a particular ECMP path. In addition, different Source Addresses or different Source UDP ports can be used for this purpose. As specified in [RFC6437], 3-tuple of Flow Label, Source Address and Destination Address fields in the IPv6 header can also be used.

- For SR-MPLS, entropy label [RFC6790] in the PM probe messages can be used.

- For SRv6, Flow Label in SRH [I-D.6man-segment-routing-header] of the PM probe messages can be used.

6. Sequence Number TLV

The message formats for DM and LM [RFC6374] do not contain sequence number for probe query packets. Sequence numbers can be useful when some probe query messages are lost or they arrive out of order.

[RFC6374] defines DM and LM probe query and response messages that can include one or more optional TLVs. New TLV Type (value TBA7) is defined in this document to carry sequence number for probe query and response messages for delay and loss measurement. The format of the

Sequence Number TLV is shown in Figure 10:

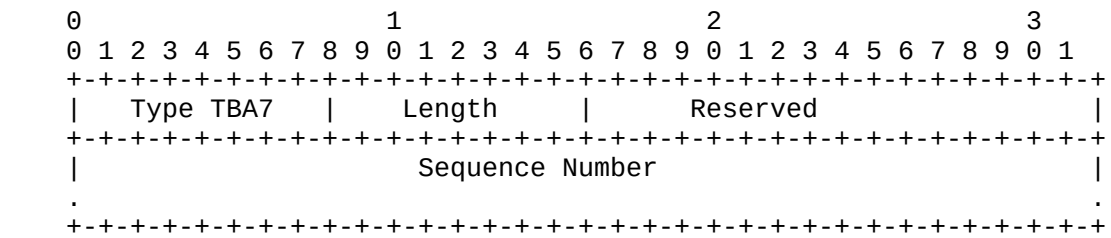


Figure 10: Sequence Number TLV

The sequence numbers start with 0 and are incremented by one for each subsequent probe query packet. The sequence number can be of any length determined by the querier node. The Sequence Number TLV is optional. The PM querier node SHOULD only insert one Sequence Number TLV in the probe query message and the responder node in the probe response message SHOULD return the first Sequence Number TLV from the probe query message and ignore other Sequence Number TLVs if present.

7. Security Considerations

The performance measurement is intended for deployment in well-managed private and service provider networks. The security considerations described in Section 8 of [RFC6374] are applicable to this specification, and particular attention should be paid to the last two paragraphs. Cryptographic measures may be enhanced by the correct configuration of access-control lists and firewalls.

8. IANA Considerations

IANA is requested to allocate following UDP ports for performance measurements:

- UDP Port TBA1: Delay Performance Measurement
- UDP Port TBA2: Loss Performance Measurement

IANA is also requested to allocate values for the following Return Path Segment List TLV Types for RFC 6374 to be carried in PM probe query messages:

- Type TBA3: SR-MPLS Label Stack of the Reverse SR Policy

- Type TBA4: SRv6 Segment List of the Reverse SR Policy
- Type TBA5: SR-MPLS Binding SID of the Reverse SR Policy
- Type TBA6: SRv6 Binding SID of the Reverse SR Policy

IANA is also requested to allocate a value for the following Sequence Number TLV Type for RFC 6374 to be carried in the PM probe query and response messages for delay and loss measurement:

- Type TBA7: Sequence Number TLV

IANA is also requested to allocate a value for the following Block Number TLV Type for RFC 6374 to be carried in the PM probe query and response messages for loss measurement:

- Type TBA8: Block Number TLV

9. References

9.1. Normative References

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

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

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS networks', RFC 6374, September 2011.

[RFC7876] Bryant, S., Sivabalan, S., and Soni, S., "UDP Return Path for Packet Loss and Delay Measurement for MPLS Networks", RFC 7876, July 2016.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017.

9.2. Informative References

[IEEE1588] IEEE, "1588-2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", March 2008.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.

[RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, November 2011.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

[RFC8029] Kompella, K., Swallow, G., Pignataro, C., Kumar, N., Aldrin, S. and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, March 2017.

[RFC8321] Fioccola, G. Ed., "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, January 2018.

[I-D.spring-segment-routing-policy] Filsfils, C., et al., "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy, work in progress.

[I-D.spring-sr-p2mp-policy] Voyer, D. Ed., et al., "SR Replication Policy for P2MP Service Delivery", draft-voyer-spring-sr-p2mp-policy, work in progress.

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

[I-D.spring-sr-mpls-pm] Filsfils, C., Gandhi, R. Ed., et al. "Performance Measurement in Segment Routing Networks with MPLS Data Plane", draft-gandhi-spring-sr-mpls-pm, work in progress.

[I-D.pce-binding-label-sid] Filsfils, C., et al., "Carrying Binding

Label Segment-ID in PCE-based Networks",
draft-sivabalan-pce-binding-label-sid, work in progress.

[I-D.spring-mpls-path-segment] Cheng, W., et al., "Path Segment in MPLS Based Segment Routing Network", draft-cheng-spring-mpls-path-segment, work in progress.

[I-D.pce-sr-path-segment] Li, C., et al., "Path Computation Element Communication Protocol (PCEP) Extension for Path Identification in Segment Routing (SR)", draft-li-pce-sr-path-segment, work in progress.

[I-D.ippm-stamp] Mirsky, G. et al. "Simple Two-way Active Measurement Protocol", draft-ietf-ippm-stamp, work in progress.

[BBF.TR-390] "Performance Measurement from IP Edge to Customer Equipment using TWAMP Light", BBF TR-390, May 2017.

Acknowledgments

The authors would like to thank Nagendra Kumar and Carlos Pignataro for the discussion on SRv6 Performance Measurement.

Contributors

Sagar Soni
Cisco Systems, Inc.
Email: sagsoni@cisco.com

Patrick Khordoc
Cisco Systems, Inc.
Email: pkhordoc@cisco.com

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

Daniel Bernier
Bell Canada
Email: daniel.bernier@bell.ca

Dirk Steinberg
Steinberg Consulting
Germany
Email: dws@dirksteinberg.de

Authors' Addresses

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

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

Daniel Voyer

Bell Canada
Email: daniel.voyer@bell.ca

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy
Email: stefano.salsano@uniroma2.it

Pier Luigi Ventre
CNIT
Italy
Email: pierluigi.ventre@cnit.it

Mach(Guoyi) Chen
Huawei
Email: mach.chen@huawei.com

SFC
Internet-Draft
Intended status: Informational
Expires: December 20, 2018

J. Guichard, Ed.
H. Song
Huawei
J. Tantsura
Nuage Networks
J. Halpern
Ericsson
W. Henderickx
Nokia
M. Boucadair
Orange
June 18, 2018

NSH and Segment Routing Integration for Service Function Chaining (SFC)
draft-guichard-sfc-nsh-sr-02

Abstract

This document describes two application scenarios where Network Service Header (NSH) and Segment Routing (SR) techniques can be deployed together to support Service Function Chaining (SFC) in an efficient manner while maintaining separation of the service and transport planes as originally intended by the SFC architecture.

In the first scenario, an NSH-based SFC is created using SR as the transport between SFFs. SR in this case is just one of many encapsulations that could be used to maintain the transport-independent nature of NSH-based service chains.

In the second scenario, SR is used to represent each service hop of the NSH-based SFC as a segment within the segment-list. SR and NSH in this case are integrated.

In both scenarios SR is responsible for steering packets between SFFs along a given SFP while NSH is responsible for maintaining the integrity of the service plane, the SFC instance context, and any associated metadata.

These application scenarios demonstrate that NSH and SR can work jointly and complement each other leaving the network operator with the flexibility to use whichever transport technology makes sense in specific areas of their network infrastructure, and still maintain an end-to-end service plane using NSH.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. SFC Overview and Rationale

The dynamic enforcement of a service-derived, adequate forwarding policy for packets entering a network that supports advanced Service Functions (SFs) has become a key challenge for operators and service providers. Particularly, cascading SFs, for example at the Gi interface in the context of mobile network infrastructure, have shown their limits, such as the same redundant classification features must be supported by many SFs in order to execute their function, some SFs are receiving traffic that they are not supposed to process (e.g., TCP proxies receiving UDP traffic), which inevitably affects their dimensioning and performance, an increased design complexity related to the properly ordered invocation of several SFs, etc.

In order to solve those problems and to avoid the adherence with the underlying physical network topology while allowing for simplified service delivery, Service Function Chaining (SFC) techniques have been introduced.

SFC techniques are meant to rationalize the service delivery logic and master the companion complexity while optimizing service activation time cycles for operators that need more agile service delivery procedures to better accommodate ever-demanding customer requirements. Indeed, SFC allows to dynamically create service planes that can be used by specific traffic flows. Each service plane is realized by invoking and chaining the relevant service functions in the right sequence. [RFC7498] provides an overview of the SFC problem space and [RFC7665] specifies an SFC architecture. The SFC architecture has the merit to not make assumptions on how advanced features (e.g., load-balancing, loose or strict service paths) have to be enabled with a domain. Various deployment options are made available to operators with the SFC architecture and this approach is fundamental to accommodate various and heterogeneous deployment contexts.

Many approaches can be considered for encoding the information required for SFC purposes (e.g., communicate a service chain pointer, encode a list of loose/explicit paths, disseminate a service chain identifier together with a set of context information, etc.). Likewise, many approaches can also be considered for the channel to be used to carry SFC-specific information (e.g., define a new header, re-use existing fields, define an IPv6 extension header, etc.). Among all these approaches, the IETF endorsed a transport-independent SFC encapsulation scheme: NSH [RFC8300]; which is the most mature SFC

encapsulation solution. This design is pragmatic as it does not require replicating the same specification effort as a function of underlying transport encapsulation. Moreover, this design approach encourages consistent SFC-based service delivery in networks enabling distinct transport protocols in various segments of the network or even between SFFs vs SF-SFF hops.

1.2. SFC within SR Networks

As described in [I-D.ietf-spring-segment-routing], Segment Routing (SR) leverages the source routing technique. Concretely, a node steers a packet through an SR policy instantiated as an ordered list of instructions called segments. While initially designed for policy-based source routing, SR also finds its application in supporting SFC [I-D.xu-clad-spring-sr-service-chaining]. The two SR flavors, namely MPLS-SR [I-D.ietf-spring-segment-routing-mpls] and SRv6 [I-D.ietf-6man-segment-routing-header], can both encode a Service Function (SF) as a segment so that an SFC can be specified as a segment list. Nevertheless, and as discussed in [RFC7498], traffic steering is only a subset of the issues that motivated the design of the SFC architecture. Further considerations such as simplifying classification at intermediate SFs and allowing for coordinated behaviors among SFs by means of supplying context information should be taken into account when designing an SFC data plane solution.

While each scheme (i.e., NSH-based SFC and SR-based SFC) can work independently, this document describes how the two can be used together in concert and complement each other through two representative application scenarios. Both application scenarios may be supported using either MPLS-SR or SRv6:

- NSH-based SFC with SR-based transport: in this scenario segment routing provides the transport encapsulation between SFFs while NSH is used to convey and trigger SFC polices.

- SR-based SFC with integrated NSH service plane: in this scenario each service hop of the SFC is represented as a segment of the SR segment-list. SR is responsible for steering traffic through the necessary SFFs as part of the segment routing path and NSH is responsible for maintaining the service plane, and holding the SFC instance context and associated metadata.

It is of course possible to combine both of these two scenarios so as to support specific deployment requirements and use cases.

2. NSH-based SFC with SR-based transport tunnel

Because of the transport-independent nature of NSH-based service chains, it is expected that the NSH has broad applicability across different domains of a network. By way of illustration the various SFs involved in a service chain are available in a single data center, or spread throughout multiple locations (e.g., data centers, different POPs), depending upon the operator preference and/or availability of service resources. Regardless of where the service resources are deployed it is necessary to provide traffic steering through a set of SFFs and NSH-based service chains provide the flexibility for the network operator to choose which particular transport encapsulation to use between SFFs, which may be different depending upon which area of the network the SFFs/SFs are currently deployed. Therefore from an SFC architecture perspective, segment routing is simply one of multiple available transport encapsulations that can be used for traffic steering between SFFs. Concretely, NSH does not require to use a unique transport encapsulation when traversing a service chain. NSH-based service forwarding relies upon underlying service node capabilities.

The following three figures provide an example of an SFC established for flow F that has SF instances located in different data centers, DC1 and DC2. For the purpose of illustration, let the SFC’s Service Path Identifier (SPI) be 100 and the initial Service Index (SI) be 255.

Referring to Figure 1, packets of flow F in DC1 are classified into an NSH-based SFC and encapsulated after classification as <Inner Pkt><NSH: SPI 100, SI 255><Outer-transport> and forwarded to SFF1 (which is the first SFF hop for this service chain).

After removing the outer transport encapsulation, that may or may not be MPLS-SR or SRv6, SFF1 uses the SPI and SI carried within the NSH encapsulation to determine that it should forward the packet to SF1. SF1 applies its service, decrements the SI by 1, and returns the packet to SFF1. SFF1 therefore has <SPI 100, SI 254> when the packet comes back from SF1. SFF1 does a lookup on <SPI 100, SI 254> which results in <next-hop: DC1-GW1> and forwards the packet to DC1-GW1.

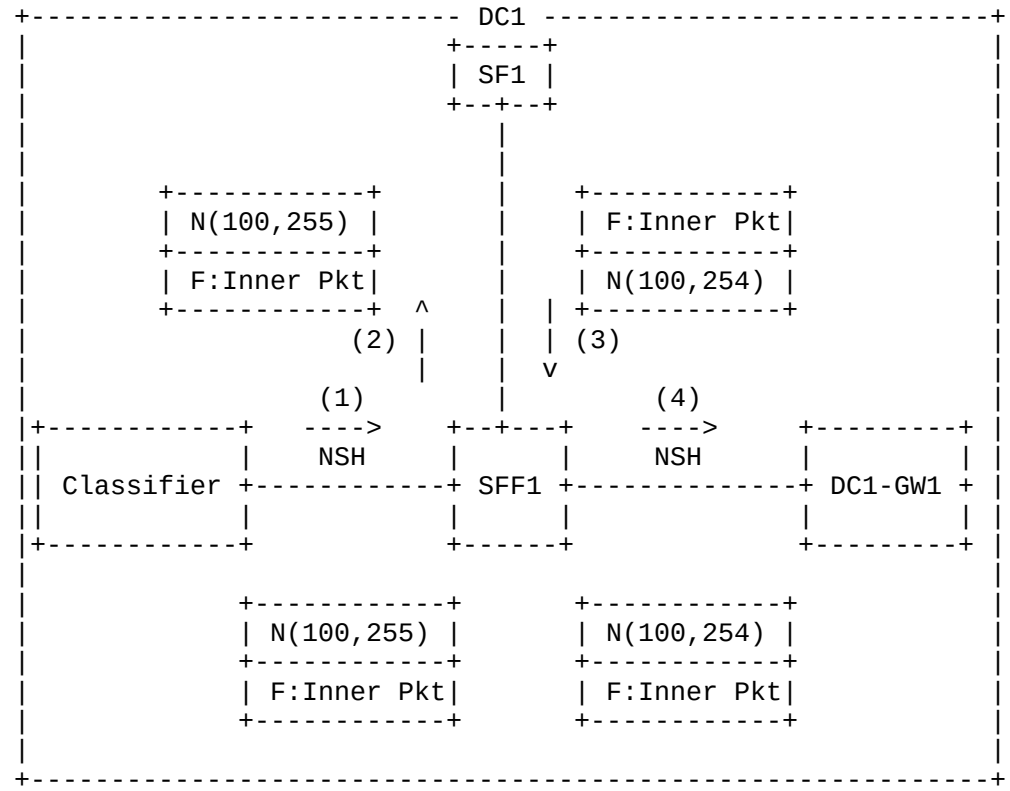


Figure 1: SR for inter-DC SFC - Part 1

Referring now to Figure 2, DC1-GW1 performs a lookup on the information conveyed in the NSH which results in <next-hop: DC2-GW1, encapsulation: SR>. The SR encapsulation has the SR segment-list to forward the packet across the inter-DC network to DC2.

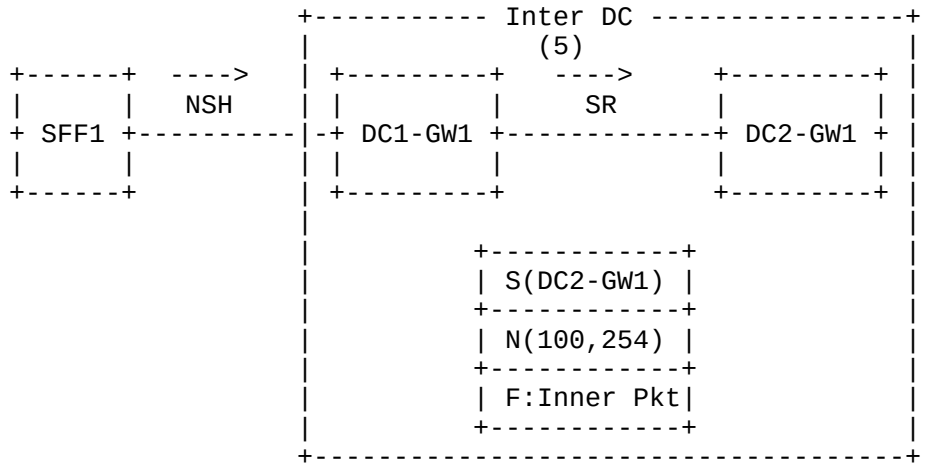


Figure 2: SR for inter-DC SFC - Part 2

When the packet arrives at DC2, as shown in Figure 3, the SR encapsulation is removed and DC2-GW1 performs a lookup on the NSH which results in next-hop: SFF2. The outer transport encapsulation may be any transport that is able to identify NSH as the next protocol.

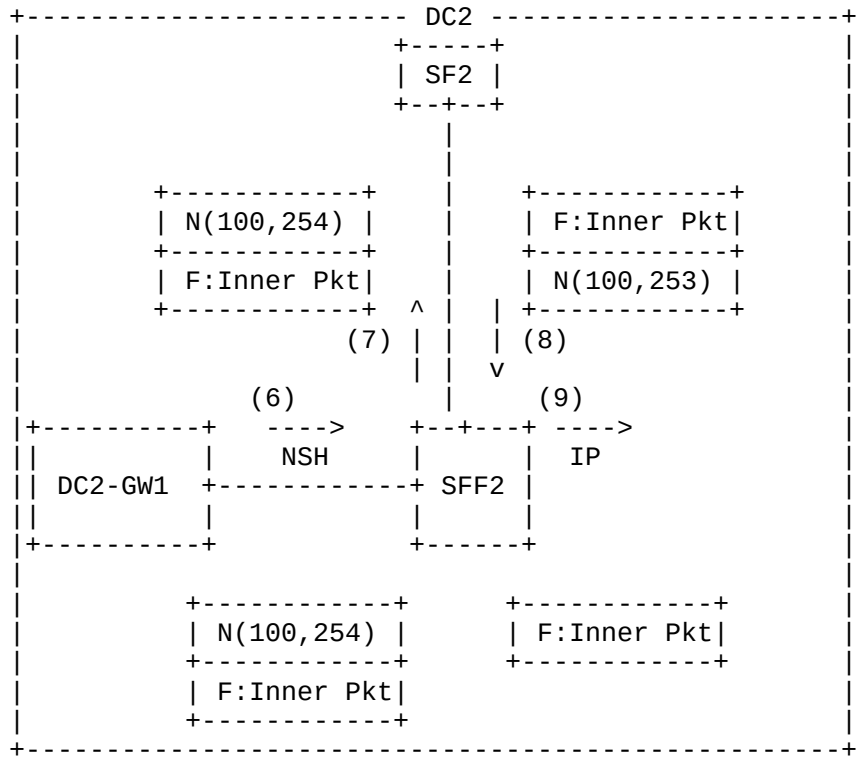


Figure 3: SR for inter-DC SFC - Part 3

The benefits of this scheme are listed hereafter:

- The network operator is able to take advantage of the transport-independent nature of the NSH encapsulation.
- The network operator is able to take advantage of the traffic steering capability of SR where appropriate.
- Light-weight NSH is used in the data center for SFC and avoids more complex hierarchical SFC schemes between data centers.
- Clear responsibility division and scope between NSH and SR.

Note that this scenario is applicable to any case where multiple segments of a service chain are distributed into multiple domains or where traffic-engineered paths are necessary between SFFs (strict forwarding paths for example).

3. SR-based SFC with Integrated NSH Service Plane

In this scenario we assume that the SFs are NSH-aware and therefore it should not be necessary to implement an SFC proxy to achieve Service Function Chaining. The operation relies upon SR to perform SFF-SFF transport and NSH to provide the service plane between SFs thereby maintaining SFC context and metadata.

When a service chain is established, a packet associated with that chain will first encapsulate an NSH that will be used to maintain the end-to-end service plane through use of the SFC context. The SFC context (e.g., the service plane path referenced by the SPI) is used by an SFF to determine the SR segment list for forwarding the packet to the next-hop SFFs. The packet is then encapsulated using the (transport-specific) SR header and forwarded in the SR domain following normal SR operation.

When a packet has to be forwarded to an SF attached to an SFF, the SFF strips the SR information of the packet, updates the SR information, and saves it to a cache indexed by the NSH SPI. This saved SR information is used to encapsulate and forward the packet(s) coming back from the SF.

When the SF receives the packet, it processes it as usual and sends it back to the SFF. Once the SFF receives this packet, it extracts the SR information using the NSH SPI as the index into the cache. The SFF then pushes the SR header on top of the NSH header, and forwards the packet to the next segment in the segment list.

Figure 4 illustrates an example of this scenario.

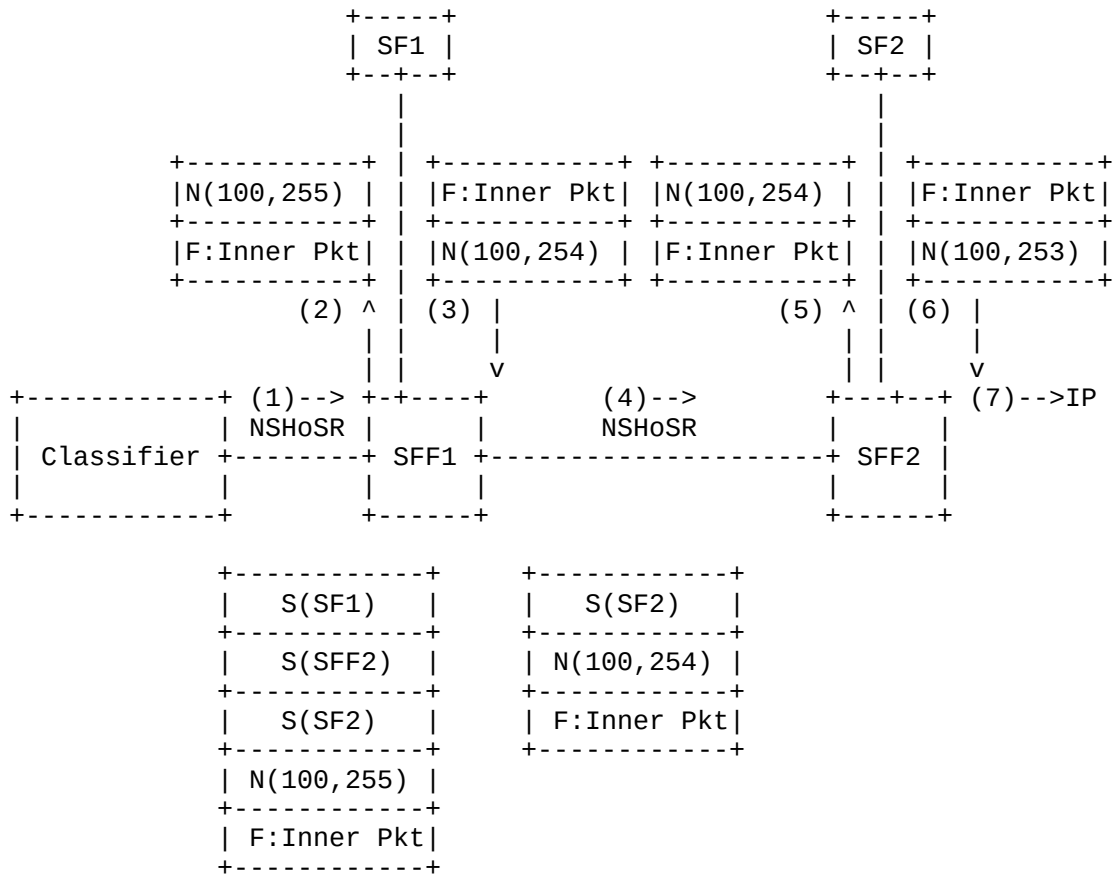


Figure 4: NSH over SR for SFC

The benefits of this scheme include:

- It is economically sound for SF vendors to only support one unified SFC solution. The SF is unaware of the SR.
- It simplifies the SFF (i.e., the SR router) by nullifying the needs for re-classification and SR proxy.
- It provides a unique and standard way to pass metadata to SFs. Note that currently there is no solution for MPLS-SR to carry metadata and there is no solution to pass metadata to SR-unaware SFs.
- SR is also used for forwarding purposes including between SFFs.

- It takes advantage of SR to eliminate the NSH forwarding state in SFFs. This applies each time strict or loose SFPs are in use.

- It requires no interworking as would be the case if MPLS-SR based SFC and NSH-based SFC were deployed as independent mechanisms in different parts of the network.

4. Encapsulation Details

4.1. NSH using MPLS-SR Transport

MPLS-SR instantiates Segment IDs (SIDs) as MPLS labels and therefore the segment routing header is a stack of MPLS labels.

When carrying NSH within an MPLS-SR transport, the full encapsulation headers are as illustrated in Figure 5.

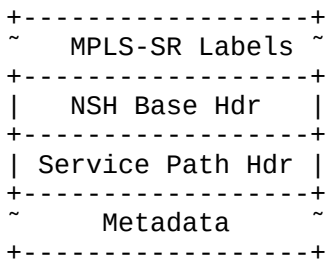


Figure 5: NSH using MPLS-SR Transport

As described in [I-D.ietf-spring-segment-routing] the IGP signaling extension for IGP-Prefix segment includes a flag to indicate whether directly connected neighbors of the node on which the prefix is attached should perform the NEXT operation or the CONTINUE operation when processing the SID. When NSH is carried beneath MPLS-SR it is necessary to terminate the NSH-based SFC at the tail-end node of the MPLS-SR label stack. This is the equivalent of MPLS Ultimate Hop Popping (UHP) and therefore the prefix-SID associated with the tail-end of the SFC MUST be advertised with the CONTINUE operation so that the penultimate hop node does not pop the top label of the MPLS-SR label stack and thereby expose NSH to the wrong SFF. It is RECOMMENDED that a specific prefix-SID be allocated at each node for use by the SFC application for this purpose.

At the end of the MPLS-SR path it is necessary to provide an indication to the tail-end that NSH follows the MPLS-SR label stack.

There are several ways to achieve this but its specification is outside the scope of this document.

4.2. NSH using SRv6 Transport

When carrying NSH within an SRv6 transport the full encapsulation is as illustrated in Figure 6.

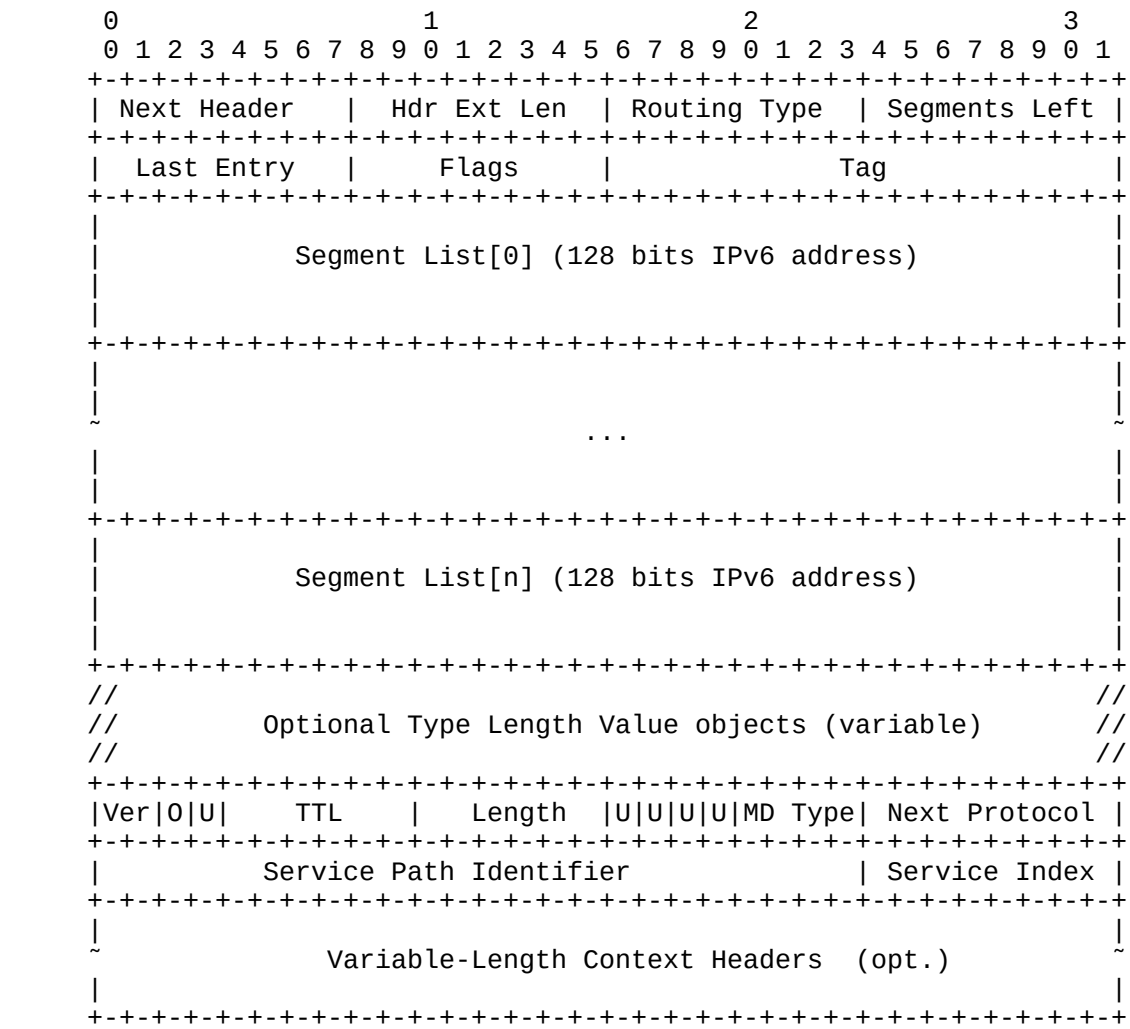


Figure 6: NSH using SRv6 Transport

5. Security Considerations

Generic SFC-related security considerations are discussed in [RFC7665]. NSH-specific security considerations are discussed in [RFC8300].

6. IANA Considerations

This memo includes no request to IANA.

7. Acknowledgments

TBD.

8. References

8.1. Normative References

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[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-12 (work in progress), February 2018.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

8.2. Informative References

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Raza, K., Dukes, D., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-09 (work in progress), March 2018.

[I-D.xu-clad-spring-sr-service-chaining]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca, d., Decraene, B., Yadlapalli, C., Henderickx, W., Salsano, S., and S. Ma, "Segment Routing for Service Chaining", draft-xu-clad-spring-sr-service-chaining-00 (work in progress), December 2017.

[RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <https://www.rfc-editor.org/info/rfc7498>.

Authors' Addresses

James N Guichard (editor)
Huawei
2330 Central Express Way
Santa Clara
USA

Email: james.n.guichard@huawei.com

Haoyu Song
Huawei
2330 Central Express Way
Santa Clara
USA

Email: haoyu.song@huawei.com

Jeff Tantsura
Nuage Networks
USA

Email: jefftant.ietf@gmail.com

Joel Halpern
Ericsson
USA

Email: joel.halpern@ericsson.com

Wim Henderickx
Nokia
USA

Email: wim.henderickx@nokia.com

Mohamed Boucadair
Orange
USA

Email: mohamed.boucadair@orange.com

Routing area
Internet-Draft
Intended status: Informational
Expires: April 20, 2019

S. Hegde
C. Bowers
Juniper Networks Inc.
S. Litkowski
Orange
X. Xu
Alibaba Inc.
F. Xu
Tencent

October 17, 2018

Node Protection for SR-TE Paths
draft-hegde-spring-node-protection-for-sr-te-paths-04

Abstract

Segment routing supports the creation of explicit paths using adjacency-sids, node-sids, and binding-sids. It is important to provide fast reroute (FRR) mechanisms to respond to failures of links and nodes in the Segment-Routed Traffic-Engineered(SR-TE) path. A point of local repair (PLR) can provide FRR protection against the failure of a link in an SR-TE path by examining only the first (top) label in the SR label stack. In order to protect against the failure of a node, a PLR may need to examine the second label in the stack as well in order to determine SR-TE path beyond the failed node. This document specifies how a PLR can use the first and second label in the label stack describing an SR-TE path to provide protection against node failures.

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

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 20, 2019.

Table of Contents

1. Introduction

It is possible for a routing device to completely go out of service abruptly due to power failure, hardware failure or software crashes. Node protection is an important property of the Fast Reroute mechanism. It provides protection against a node failure by rerouting traffic around the failed node. For example, the mechanisms described in Loop Free Alternates ([RFC5286]), Remote Loop

Free Alternates ([RFC8102]), and
[I-D.bashandy-rtgwg-segment-routing-ti-lfa] can be used to provide node protection to ensure minimal traffic loss after a node failure.

Section 2 describes problems with SR-TE paths and the need for a specialized mechanism to provide node protection for SR-TE paths. Section 3 describes the solution applied to paths built using adjacency-sids and node-sids.

2. Node Failures Along SR-TE Paths

The topology shown in Figure 1. illustrates a example network topology with SPRING enabled on each node.

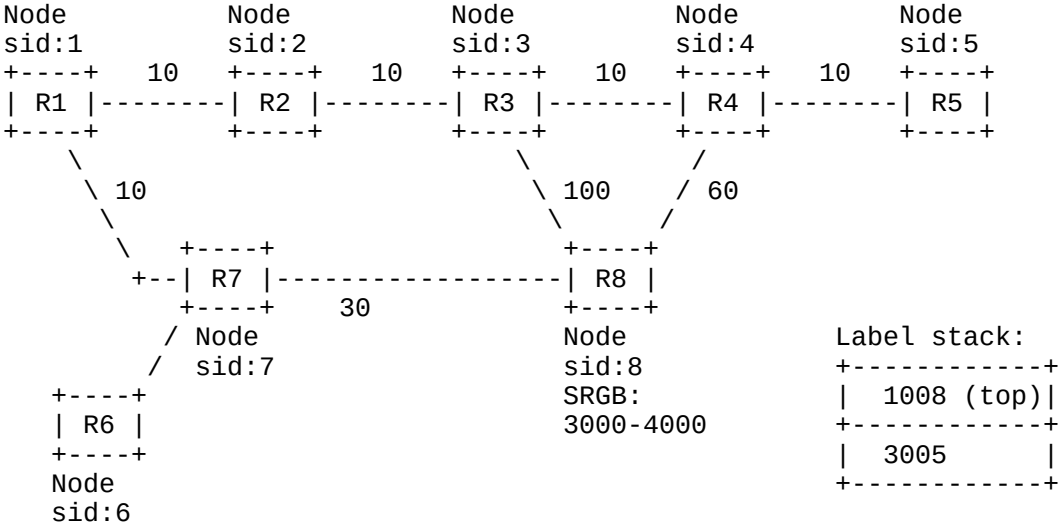


Figure 1: Example topology. The segment index for each node is shown in the diagram. All nodes have SRGB = [1000-2000], except for R8 which has SRGB = [3000-4000]. A label stack that represents the path R1->R7->R8->R4->R5 is shown as well.

2.1. Node protection for node-sid explicit paths

Consider an explicit path in the topology in Figure 1 from R1->R5 via R1->R7->R8->R4->R5. This path can be built using the shortest paths from R1-to-R8 and R8-to-R5. The label stack to instantiate this path contains two node-sids 1008 and 3005. The 1008 label will take the packet from R1 to R8 via R7 and get popped. The next label in the stack 3005 will take the packet from R8 to the destination R5 via R4. If the node R8 goes down, it is not possible for R7 to perform FRR

without examining the second label in the incoming label stack (3005).

Note that in the absence of a failure, R7 does not need to understand the meaning of the second label (3005) in order to perform normal forwarding. However, in order to support node protection, R7 will need to understand the meaning of label 3005 in order to determine where the packet is headed after R8.

2.2. Node-Protection for Anycast-SIDs

A prefix segment advertised as a node SID may only be advertised by one node in the network. Instead, an anycast prefix segment may be advertised by more than one node. In some situations, one can use anycast SIDs to construct SR-TE paths that are protected against node failure, without the need for the mechanism described in this document.

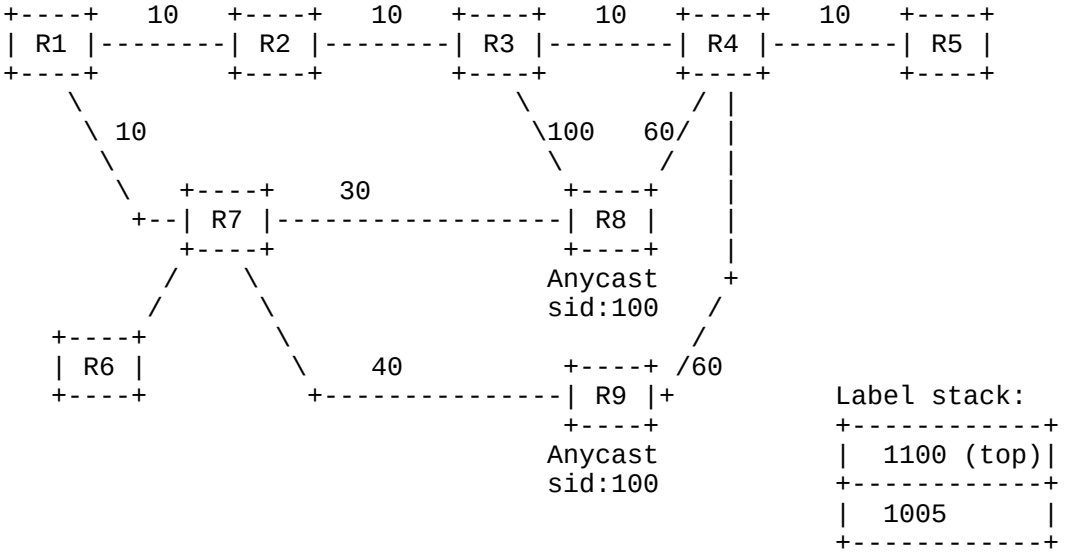


Figure 2: Topology illustrating use of anycast-sids to protect against node failures. All nodes have SRGB = [1000-2000].

An example of this is shown in Figure 2. In this example, R8 and R9 advertise an anycast SID of 100. The label stack in this example = [1100, 1005];. The top label (1100) corresponds to the anycast SID advertised by both R8 and R9. In the absence of a failure, the packet sent by R1 with this label stack will follow the path from R1->R5 along R1->R7->R8->R4->R5.

If R7 is performing a per-prefix LFA calculation [RFC5286], then R7 will install a backup next-hop to R9 for this anycast SID, protecting against the failure of the primary next-hop to R8. This backup path does not pass through R8, so it is would not be affected by a complete failure of node R8. As illustrated by this example, for some topologies node-protecting SR-TE paths can constructed through the use of anycast SIDs, as opposed to the mechanism described in this document.

2.3. Node-protection for adj-sid explicit paths

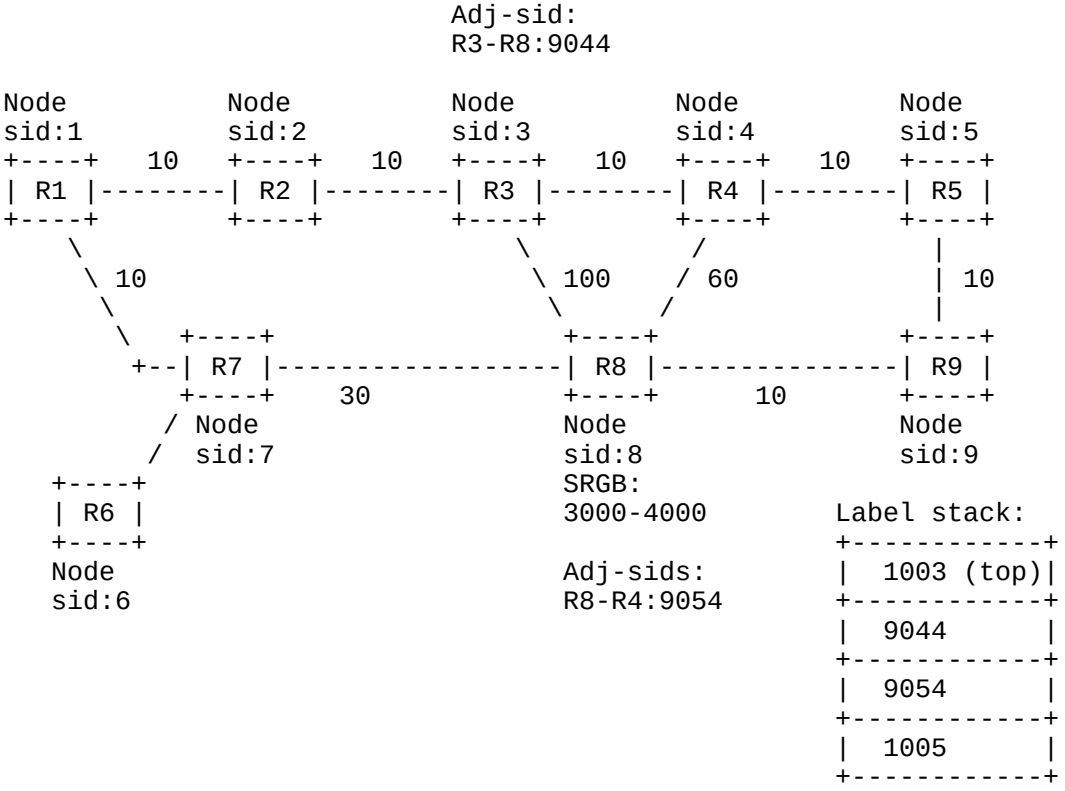


Figure 3: Explicit path using an adjacency sid. All nodes have SRGB = [1000-2000], except for R8 which has SRGB = [3000-4000].

Consider an explicit path from R1->R5 via R1->R2->R3->R8->R4->R5. This path can be built using a combination of node sids and adjacency sids, as shown in Figure 3. The diagram shows the label stack needed to instantiate this path, as well as several adjacency sids advertised by nodes involved in this path. When a packet leaving R1 with this label stack reaches R3, the top label is 9044, which will

take the packet to R8. The next-next-hop in the path is R4. To provide protection for the failure of node R8, R3 would need to send the the packet to R4 without going through R8. However, the only way R3 can learn that the packet needs to go to the R4 is to examine the next label in the stack, label 9054. Since R3 knows that R8 has advertised label 9054 as the adjacency segment for the link from R8 to R4, R3 knows that a backup path can merge back into the original explicit path at R4.

3. Detailed Solution using Context Tables

This section provides a detailed description of how to construct node-protecting backup paths for SR-TE paths using context tables. The end result of this description is externally visible forwarding behavior that can be specified as a packet arriving at a PLR with a particular incoming label stack and leaving the PLR on a particular outgoing interface with a particular outgoing label stack. There may be other methods of arriving at the same externally visible forwarding behavior as described in draft [I-D.bashandy-rtgwg-segment-routing-ti-lfa]. It is not the intent of this document to exclude other methods, as long as the externally visible forwarding behavior is the same as produced by this method.

3.1. Building Context Tables

[RFC5331] introduced the concept of Context Specific Label Spaces and there are various applications making use of this concept.A context label table on a router represents the Label Forwarding Information Base (LFIB) from the point of view of a particular neighbor . Context tables are built by constructing incoming label mappings advertised by the neighbor and the actions corresponding to those labels. The labels advertised by each node are local to the node and may not be unique across the segment routing domain. The context tables are separate tables built on a per-neighbor basis on every node to ensure they represent LFIBs of a particular neighbor.

When a PLR needs to protect an SR-TE path against the failure of a neighbor N, it creates a context table associated with N. This context table is populated with the following segment routing forwarding entries:

- All the Prefix-SIDs of the network. The programmed incoming label map uses the SRGB of N to compute the input label value. The NHLFE (Next Hop Label Forwarding Entry) is then constructed by looking into all the nexthops for the prefix-SID and choosing a loop-free path as explained in Section 3.2

- All the Adjacency SIDs advertised by N. The NHLFE is constructed as explained in Section 3.3

The following section illustrates how the context table is constructed to allow the PLR to provide node-protecting paths for the next-next hops in the topology shown in Figure 1 and Figure 3.

3.2. Node protection for node SIDs

Figure 4 shows the routing table entries on R7 corresponding to the node SIDs to reach R1 and R8 for the topology in Figure 1. In the absence of a failure, a packet with a label stack whose top label is 1008 will have its top label popped by R7 (assuming PHP behavior), and R7 will forward the packet to R8. When the interface to R8 is down, the backup next-hop entry is used. R7 will pop the top label of 1008, and use the context table that R7 computed for R8 to evaluate the next label on the stack.

R7's Routing Table (partial)
Transits routes for Node SIDs for R1 and R8

In label	Outgoing label action
1001	Primary: pop, fwd to R1
	Backup: pop, lookup context.r1
1008	Primary: pop, fwd to R8
	Backup: pop, lookup context.r8

R7's Context Table for R8 (context.r8, partial)

In label	Outgoing label action
3004	swap 1004, fwd to R1
3005	swap 1005, fwd to R1
3008	drop

Figure 4: Building node-protecting backup paths for SR-TE paths involving node SIDs

R7 builds context table for R8 using the following process. R7 computes the mapping of incoming label to node-sid that R8 expects to see based on the SRGB advertised by R8. In the example in Figure 1,

R7 can determine that R8 interprets in incoming label of 3005 as mapping to the the node SID for R5.

R7 then computes a loop-free backup path to reach R5 which is node-protecting with respect to the failure of R8. In this example, the backup path computed by R7 to reach R5 without passing through R8 can be achieved forwarding the packet to R1 with a top label of 1005, corresponding to the node SID for R5 in the context of R1's SRGB. The loop-free path computation may be based on a mechanism such as LFA, R-LFA, TI-LFA, or constraint based SPF avoiding failure. To populate the context table for R8, R7 maps the out label actions corresponding to the backup path to R5 to the incoming label 3005. This results in the entry for label 3005 showin in context.r8 in Figure 4.

Therefore, when a packet arrives at R7 with label stack = [1008, 3005], and the link from R7 to R8 has recently failed, R7 will use backup next-hop entry for label 1008 in its main routing table. Based on this entry, R7 will pop label 1008, and use context.r8 to lookup the new top label = 3005. R7 will swap label 3005 for 1005 and forward the packet to R1. This will get the packet to R5 on a node protecting backup path.

Note that R7 activates the node-protecting backup path when it detects that the link to R8 has failed. R7 does not know that node R8 has actually failed. However, the node-protecting backup path is computed assuming that the failure of the link to R8 implies that R8 has failed.

3.3. Node protection for adjacency SIDs

This section gives an example of how to constuct node-protecting backup paths when the SR-TE path uses adjacency SIDs. Figure 5 shows some of the routing table entries for R3 corresponding to the sample network shown in Figure 3. When the top label of the label stack is an adjacency SID, the PLR needs to recognize that in order to provide a node-protecting backup path, it needs to pop the top label and examine the next label in the context of the next-hop router identified by the top label adjacency SID. In this example, when R3 is constructing its routing table, it recognizes that label 9044 corresponds to a next-hop of R8, so it installs a backup entry, corresponding to the failure of the link to R8, when pops label 9044, and then examines the new top label in the context of R8.

R3's Routing Table (partial)
Transit route for Adj SID

In label	Outgoing label action
9044	Primary: pop, fwd to R8
	Backup: pop, lookup context.r8

R3's Context Table for R8 (context.r8, partial)

In label	Outgoing label action
3005	swap 1005, fwd to R4
9054	pop, fwd to R4

Figure 5: Building node-protecting backup paths for SR-TE paths involving adjacency SIDs

R3 constructs its context table for R8 by determining which labels R8 expects to receive to accomplish different forwarding actions. The entry for incoming label 3005 in context.r8 in Figure 5 corresponds to a node SID. This entry is computed using the methods described in Section 3.2

The entry for incoming label 9054 in context.r8 corresponds to an adjacency SID. R3 recognizes that R8 has advertised this adjacency SID for the link from R8 to R4 in Figure 3. So R3 determines the outgoing label action needed to reach R4 without passing through R8. This can be accomplished by popping the label 9054, and forwarding the packet directly on the link from R3 to R4.

3.4. Node protection for edge nodes

The node protection mechanism described in the previous sections depends on the assumption that the label immediately below the top label in the label stack is understood in the IGP domain. When the provider edge routers exchange service labels via BGP or some other non-IGP mechanism the bottom label is not understood in the IGP domain.

The egress node protection mechanisms described in the draft [I-D.ietf-mpls-egress-protection-framework] is applicable to this usecase and no additional changes will be required for SR based networks

4. Security Considerations

TBD

5. IANA Considerations

6. Acknowledgments

The authors would like to thank Peter Psenak and Bruno Decreane for their review and suggestions.

7. References

7.1. Normative References

[RFC5286] Atlas, A., Ed. and A. Zinin, Ed., "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, DOI 10.17487/RFC5286, September 2008, <https://www.rfc-editor.org/info/rfc5286>.

[RFC5331] Aggarwal, R., Rekhter, Y., and E. Rosen, "MPLS Upstream Label Assignment and Context-Specific Label Space", RFC 5331, DOI 10.17487/RFC5331, August 2008, <https://www.rfc-editor.org/info/rfc5331>.

7.2. Informative References

[I-D.bashandy-rtgwg-segment-routing-ti-lfa] Bashandy, A., Filsfils, C., Decraene, B., Litkowski, S., Francois, P., daniel.voyer@bell.ca, d., Clad, F., and P. Camarillo, "Topology Independent Fast Reroute using Segment Routing", draft-bashandy-rtgwg-segment-routing-ti-lfa-05 (work in progress), October 2018.

[I-D.ietf-mpls-egress-protection-framework] Shen, Y., Jeganathan, J., Decraene, B., Gredler, H., Michel, C., Chen, H., and Y. Jiang, "MPLS Egress Protection Framework", draft-ietf-mpls-egress-protection-framework-02 (work in progress), July 2018.

[I-D.ietf-spring-segment-routing] Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[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>.

[RFC8102] Sarkar, P., Ed., Hegde, S., Bowers, C., Gredler, H., and S. Litkowski, "Remote-LFA Node Protection and Manageability", RFC 8102, DOI 10.17487/RFC8102, March 2017, <https://www.rfc-editor.org/info/rfc8102>.

Authors' Addresses

Shraddha Hegde
Juniper Networks Inc.
Exora Business Park
Bangalore, KA 560103
India

Email: shraddha@juniper.net

Chris Bowers
Juniper Networks Inc.

Email: cbowers@juniper.net

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Xiaohu Xu
Alibaba Inc.
Beijing
China

Email: xiaohu.xxh@alibaba-inc.com

Feng Xu
Tencent
China

Email: oliverxu@tencent.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

C. Filsfils
S. Sivabalan, Ed.
Cisco Systems, Inc.
D. Voyer
Bell Canada
A. Bogdanov
Google, Inc.
P. Mattes
Microsoft

October 22, 2018

Segment Routing Policy Architecture
draft-ietf-spring-segment-routing-policy-02.txt

Abstract

Segment Routing (SR) allows a headend node to steer a packet flow along any path. Intermediate per-flow states are eliminated thanks to source routing. The headend node steers a flow into an SR Policy. The header of a packet steered in an SR Policy is augmented with an ordered list of segments associated with that SR Policy. This document details the concepts of SR Policy and steering into an SR Policy.

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

Table of Contents

1. Introduction

Segment Routing (SR) allows a headend node to steer a packet flow along any path. Intermediate per-flow states are eliminated thanks to source routing [RFC8402].

The headend node is said to steer a flow into an Segment Routing Policy (SR Policy).

The header of a packet steered into an SR Policy is augmented with an ordered list of segments associated with that SR Policy.

This document details the concepts of SR Policy and steering packets into an SR Policy. These apply equally to the MPLS and SRv6 instantiations of segment routing.

For reading simplicity, the illustrations are provided for the MPLS instantiations.

2. SR Policy

An SR Policy is a framework that enables instantiation of an ordered list of segments on a node for implementing a source routing policy with a specific intent for traffic steering from that node.

The Segment Routing architecture [RFC8402] specifies that any instruction can be bound to a segment. Thus, an SR Policy can be built using any type of Segment Identifier (SID) including those associated with topological or service instructions.

This section defines the key aspects and constituents of an SR Policy.

2.1. Identification of an SR Policy

An SR Policy is identified through the tuple <headend, color, endpoint>. In the context of a specific headend, one may identify an SR policy by the <color, endpoint> tuple.

The headend is the node where the policy is instantiated/implemented. The headend is specified as an IPv4 or IPv6 address and is expected to be unique in the domain.

The endpoint indicates the destination of the policy. The endpoint is specified as an IPv4 or IPv6 address and is expected to be unique in the domain. In a specific case (refer to Section 8.8.1), the endpoint can be the null address (0.0.0.0 for IPv4, ::0 for IPv6).

The color is a 32-bit numerical value that associates the SR Policy with an intent (e.g. low-latency).

The endpoint and the color are used to automate the steering of service or transport routes on SR Policies (refer to Section 8).

An implementation MAY allow assignment of a symbolic name comprising of printable ASCII characters to an SR Policy to serve as a user-friendly attribute for debug and troubleshooting purposes. Such symbolic names may identify an SR Policy when the naming scheme ensures uniqueness.

2.2. Candidate Path and Segment List

An SR Policy is associated with one or more candidate paths. A candidate path is the unit for signaling of an SR Policy to a headend via protocols like Path Computation Element (PCE) Communication Protocol (PCEP) [RFC8281] or BGP SR Policy [I-D.ietf-idr-segment-routing-te-policy].

A Segment-List represents a specific source-routed path to send traffic from the headend to the endpoint of the corresponding SR policy.

A candidate path is either dynamic or explicit.

An explicit candidate path is expressed as a Segment-List or a set of Segment-Lists.

A dynamic candidate path expresses an optimization objective and a set of constraints. The headend (potentially with the help of a PCE) computes the solution Segment-List (or set of Segment-Lists) that solves the optimization problem.

If a candidate path is associated with a set of Segment-Lists, each Segment-List is associated with a weight for weighted load balancing (refer Section 2.11 for details). The default weight is 1.

2.3. Protocol-Origin of a Candidate Path

A headend may be informed about a candidate path for an SR Policy <color, endpoint> by various means including: via configuration, PCEP [RFC8281] or BGP [I-D.ietf-idr-segment-routing-te-policy].

Protocol-Origin of a candidate path is an 8-bit value which identifies the component or protocol that originates or signals the candidate path.

The table below specifies the RECOMMENDED default values:

Value	Protocol-Origin
10	PCEP
20	BGP SR Policy
30	Via Configuration

Table 1: Protocol-origin Identifier

Implementations MAY allow modifications of these default values assigned to protocols on the headend along similar lines as a routing administrative distance. Its application in the candidate path selection is described in Section 2.9.

2.4. Originator of a Candidate Path

Originator identifies the node which provisioned or signalled the candidate path on the headend. The originator is expressed in the form of a 160 bit numerical value formed by the concatenation of the fields of the tuple <ASN, node-address> as below:

- ASN : represented as a 4 byte number.
- Node Address : represented as a 128 bit value. IPv4 addresses are encoded in the lowest 32 bits.

Its application in the candidate path selection is described in Section 2.9.

When Protocol-Origin is Via Configuration, the ASN and node address MAY be set to either the headend or the provisioning controller/node ASN and address. Default value is 0 for both AS and node address.

When Protocol-Origin is PCEP, it is the IPv4 or IPv6 address of the PCE and the AS number SHOULD be set to 0 by default when not available or known.

Protocol-Origin is BGP SR Policy, it is provided by the BGP component on the headend and is:

- the BGP Router ID and ASN of the node/controller signalling the candidate path when it has a BGP session to the headend, OR
- the BGP Router ID of the eBGP peer signalling the candidate path along with ASN of origin when the signalling is done via one or more intermediate eBGP routers, OR
- the BGP Originator ID [RFC4456] and the ASN of the node/controller when the signalling is done via one or more route-reflectors over iBGP session.

2.5. Discriminator of a Candidate Path

The Discriminator is a 32 bit value associated with a candidate path that uniquely identifies it within the context of an SR Policy from a specific Protocol-Origin as specified below:

When Protocol-Origin is Via Configuration, this is an implementation's configuration model specific unique identifier for a candidate path. Default value is 0.

When PCEP is the Protocol-Origin, the method to uniquely identify signalled path will be specified in a future PCEP document. Default value is 0.

When BGP SR Policy is the Protocol-Origin, it is the distinguisher specified in Section 2.1 of [I-D.ietf-idr-segment-routing-te-policy].

Its application in the candidate path selection is described in Section 2.9.

2.6. Identification of a Candidate Path

A candidate path is identified in the context of a single SR Policy.

A candidate path is not shared across SR Policies.

A candidate path is not identified by its Segment-List(s).

> If CP1 is a candidate path of SR Policy Pol1 and CP2 is a candidate path of SR Policy Pol2, then these two candidate paths are independent, even if they happen to have the same Segment-List. The Segment-List does not identify a candidate path. The Segment-List is an attribute of a candidate path.

The identity of a candidate path MUST be uniquely established in the context of an SR Policy <headend, color, endpoint> in order to handle add, delete or modify operations on them in an unambiguous manner regardless of their source(s).

The tuple <Protocol-Origin, originator, discriminator> uniquely identifies a candidate path.

Candidate paths MAY also be assigned or signaled with a symbolic name comprising printable ASCII characters to serve as a user-friendly attribute for debug and troubleshooting purposes. Such symbolic names MUST NOT be considered as identifiers for a candidate path.

2.7. Preference of a Candidate Path

The preference of the candidate path is used to select the best candidate path for an SR Policy. The default preference is 100.

It is RECOMMENDED that each candidate path of a given SR policy has a different preference.

2.8. Validity of a Candidate Path

A candidate path is usable when it valid. A common path validity criterion is the reachability of its constituent SIDs. The validation rules are specified in Section 5.

2.9. Active Candidate Path

A candidate path is selected when it is valid and it is determined to be the best path of the SR Policy. The selected path is referred to as the "active path" of the SR policy in this document.

Whenever a new path is learned or an active path is deleted, the validity of an existing path changes or an existing path is changed, the selection process MUST be re-executed.

The candidate path selection process operates on the candidate path Preference. A candidate path is selected when it is valid and it has the highest preference value among all the candidate paths of the SR Policy.

In the case of multiple valid candidate paths of the same preference, the tie-breaking rules are evaluated on the identification tuple in the following order until only one valid best path is selected:

1. Higher value of Protocol-Origin is selected.

2. If specified by configuration, prefer the existing installed path.

3. Lower value of originator is selected.

4. Finally, the higher value of discriminator is selected.

The rules are framed with multiple protocols and sources in mind and hence may not follow the logic of a single protocol (e.g. BGP best path selection). The motivation behind these rules are as follows:

- The Protocol-Origin allows an operator to setup a default selection mechanism across protocol sources, e.g., to prefer configured over paths signalled via BGP SR Policy or PCEP.

- The preference, being the first tiebreaker, allows an operator to influence selection across paths thus allowing provisioning of multiple path options, e.g., CP1 is preferred and if it becomes invalid then fall-back to CP2 and so on. Since preference works across protocol sources it also enables (where necessary) selective override of the default protocol-origin preference, e.g., to prefer a path signalled via BGP SR Policy over what is configured.

- The originator allows an operator to have multiple redundant controllers and still maintain a deterministic behaviour over which of them are preferred even if they are providing the same candidate paths for the same SR policies to the headend.

- The discriminator performs the final tiebreaking step to ensure a deterministic outcome of selection regardless of the order in which candidate paths are signalled across multiple transport channels or sessions.

[I-D.filsfils-spring-sr-policy-considerations] provides a set of examples to illustrate the active candidate path selection rules.

2.10. Validity of an SR Policy

An SR Policy is valid when it has at least one valid candidate path.

2.11. Instantiation of an SR Policy in the Forwarding Plane

A valid SR Policy is instantiated in the forwarding plane.

Only the active candidate path SHOULD be used for forwarding traffic that is being steered onto that policy.

If a set of Segment-Lists is associated with the active path of the policy, then the steering is per flow and W-ECMP based according to the relative weight of each Segment-List.

The fraction of the flows associated with a given Segment-List is w/ Sw where w is the weight of the Segment-List and Sw is the sum of the weights of the Segment-Lists of the selected path of the SR Policy.

The accuracy of the weighted load-balancing depends on the platform implementation.

2.12. Priority of an SR Policy

Upon topological change, many policies could be recomputed or revalidated. An implementation MAY provide a per-policy priority configuration. The operator MAY set this field to indicate order in which the policies should be re-computed. Such a priority is represented by an integer in the range (0, 255) where the lowest value is the highest priority. The default value of priority is 128.

An SR Policy may comprise multiple Candidate Paths received from the same or different sources. A candidate path MAY be signaled with a priority value. When an SR Policy has multiple candidate paths with distinct signaled non-default priority values, the SR Policy as a whole takes the lowest value (i.e. the highest priority) amongst these signaled priority values.

2.13. Summary

In summary, the information model is the following:

```
          SR policy POL1 <headend, color, endpoint>
            Candidate-path CP1 <protocol-origin = 20, originator =
100:1.1.1.1, discriminator = 1>
```

```
              Preference 200
              Weight W1, SID-List1 <SID11...SID1i>
              Weight W2, SID-List2 <SID21...SID2j>
            Candidate-path CP2 <protocol-origin = 20, originator =
100:2.2.2.2, discriminator = 2>
              Preference 100
              Weight W3, SID-List3 <SID31...SID3i>
              Weight W4, SID-List4 <SID41...SID4j>
```

The SR Policy POL1 is identified by the tuple <headend, color, endpoint>. It has two candidate paths CP1 and CP2. Each is identified by a tuple <protocol-origin, originator, discriminator>. CP1 is the active candidate path (it is valid and it has the highest preference). The two Segment-Lists of CP1 are installed as the forwarding instantiation of SR policy Pol1. Traffic steered on Pol1 is flow-based hashed on Segment-List <SID11...SID1i> with a ratio W1/(W1+W2).

3. Segment Routing Database

An SR headend maintains the Segment Routing Database (SR-DB). The SR-DB is a conceptual database to illustrate the various pieces of information and their sources that may help in SR Policy computation and validation. There is no specific requirement for an implementation to create a new database as such.

An SR headend leverages the SR-DB to validate explicit candidate paths and compute dynamic candidate paths.

The information in the SR-DB MAY include:

- IGP information (topology, IGP metrics based on ISIS [RFC1195] and OSPF [RFC2328] [RFC5340])
- Segment Routing information (such as SRGB, SRLB, Prefix-SIDs, Adj-SIDs, BGP Peering SID, SRv6 SIDs) [RFC8402] [I-D.ietf-idr-bgpls-segment-routing-epe] [I-D.filsfils-spring-srv6-network-programming]
- TE Link Attributes (such as TE metric, SRLG, attribute-flag, extended admin group) [RFC5305] [RFC3630].
- Extended TE Link attributes (such as latency, loss) [RFC7810] [RFC7471]
- Inter-AS Topology information [I-D.ietf-idr-bgpls-segment-routing-epe].

The attached domain topology MAY be learned via IGP, BGP-LS or NETCONF.

A non-attached (remote) domain topology MAY be learned via BGP-LS or NETCONF.

In some use-cases, the SR-DB may only contain the attached domain topology while in others, the SR-DB may contain the topology of multiple domains and in this case it is multi-domain capable.

The SR-DB MAY also contain the SR Policies instantiated in the network. This can be collected via BGP-LS [I-D.ietf-idr-te-lsp-distribution] or PCEP [RFC8231] and [I-D.sivabalan-pce-binding-label-sid]. This information allows to build an end-to-end policy on the basis of intermediate SR policies (see Section 6 for further details).

The SR-DB MAY also contain the Maximum SID Depth (MSD) capability of nodes in the topology. This can be collected via ISIS [I-D.ietf-isis-segment-routing-msd], OSPF [I-D.ietf-ospf-segment-routing-msd], BGP-LS [I-D.ietf-idr-bgp-ls-segment-routing-msd] or PCEP [I-D.ietf-pce-segment-routing].

The use of the SR-DB for computation and validation of SR Policies is outside the scope of this document. Some implementation aspects related to this are covered in [I-D.filsfils-spring-sr-policy-considerations].

4. Segment Types

A Segment-List is an ordered set of segments represented as <S1, S2, ... Sn> where S1 is the first segment.

Based on the desired dataplane, either the MPLS label stack or the SRv6 SRH is built from the Segment-List. However, the Segment-List itself can be specified using different segment-descriptor types and the following are currently defined:

Type 1: SR-MPLS Label:
A MPLS label corresponding to any of the segment types defined for SR-MPLS (as defined in [RFC8402] or other SR-MPLS specifications) can be used. Additionally, reserved labels like explicit-null or in general any MPLS label may also be used. E.g. this type can be used to specify a label representation which maps to an optical transport path on a packet transport node. This type does not require the headend to perform SID resolution.

Type 2: SRv6 SID:

An IPv6 address corresponding to any of the segment types defined for SRv6 (as defined in [I-D.filsfils-spring-srv6-network-programming] or other SRv6 specifications) can be used. This type does not require the headend to perform SID resolution.

Type 3: IPv4 Prefix with optional SR Algorithm:
The headend is required to resolve the specified IPv4 Prefix Address to the SR-MPLS label corresponding to a Prefix SID segment (as defined in [RFC8402]). The SR algorithm (refer to Section 3.1.1 of [RFC8402]) to be used MAY also be provided.

Type 4: IPv6 Global Prefix with optional SR Algorithm for SR-MPLS:
In this case the headend is required to resolve the specified IPv6 Global Prefix Address to the SR-MPLS label corresponding to its Prefix SID segment (as defined in [RFC8402]). The SR Algorithm (refer to Section 3.1.1 of [RFC8402]) to be used MAY also be provided.

Type 5: IPv4 Prefix with Local Interface ID:
This type allows identification of Adjacency SID (as defined in [RFC8402]) or BGP EPE Peering SID (as defined in [I-D.ietf-idr-bgpls-segment-routing-epe]) label for point-to-point links including IP unnumbered links. The headend is required to resolve the specified IPv4 Prefix Address to the Node originating it and then use the Local Interface ID to identify the point-to-point link whose adjacency is being referred to. The Local Interface ID link descriptor follows semantics as specified in [RFC7752]. This type can also be used to indicate indirection into a layer 2 interface (i.e. without IP address) like a representation of an optical transport path or a layer 2 Ethernet port or circuit at the specified node.

Type 6: IPv4 Addresses for link endpoints as Local, Remote pair:
This type allows identification of Adjacency SID (as defined in [RFC8402]) or BGP EPE Peering SID (as defined in [I-D.ietf-idr-bgpls-segment-routing-epe]) label for links. The headend is required to resolve the specified IPv4 Local Address to the Node originating it and then use the IPv4 Remote Address to identify the link adjacency being referred to. The Local and Remote Address pair link descriptors follows semantics as specified in [RFC7752].

Type 7: IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SR-MPLS:
This type allows identification of Adjacency SID (as defined in [RFC8402]) or BGP EPE Peering SID (as defined in

[I-D.ietf-idr-bgpls-segment-routing-epe]) label for links including those with only Link Local IPv6 addresses. The headend is required to resolve the specified IPv6 Prefix Address to the Node originating it and then use the Local Interface ID to identify the point-to-point link whose adjacency is being referred to. For other than point-to-point links, additionally the specific adjacency over the link needs to be resolved using the Remote Prefix and Interface ID. The Local and Remote pair of Prefix and Interface ID link descriptor follows semantics as specified in [RFC7752]. This type can also be used to indicate indirection into a layer 2 interface (i.e. without IP address) like a representation of an optical transport path or a layer 2 Ethernet port or circuit at the specified node.

Type 8: IPv6 Addresses for link endpoints as Local, Remote pair for SR-MPLS:
: This type allows identification of Adjacency SID (as defined in [RFC8402]) or BGP EPE Peering SID (as defined in [I-D.ietf-idr-bgpls-segment-routing-epe]) label for links with Global IPv6 addresses. The headend is required to resolve the specified Local IPv6 Address to the Node originating it and then use the Remote IPv6 Address to identify the link adjacency being referred to. The Local and Remote Address pair link descriptors follows semantics as specified in [RFC7752].

Type 9: IPv6 Global Prefix with optional SR Algorithm for SRv6:
: The headend is required to resolve the specified IPv6 Global Prefix Address to the SRv6 END function SID (as defined in [I-D.filsfils-spring-srv6-network-programming]) corresponding to the node which is originating the prefix. The SR Algorithm (refer to Section 3.1.1 of [RFC8402]) to be used MAY also be provided.

Type 10: IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SRv6:
: This type allows identification of SRv6 END.X SID (as defined in [I-D.filsfils-spring-srv6-network-programming]) for links with only Link Local IPv6 addresses. The headend is required to resolve the specified IPv6 Prefix Address to the Node originating it and then use the Local Interface ID to identify the point-to-point link whose adjacency is being referred to. For other than point-to-point links, additionally the specific adjacency needs to be resolved using the Remote Prefix and Interface ID. The Local and Remote pair of Prefix and Interface ID link descriptor follows semantics as specified in [RFC7752].

Type 11: IPv6 Addresses for link endpoints as Local, Remote pair for SRv6:

This type allows identification of SRv6 END.X SID (as defined in [I-D.filsfils-spring-srv6-network-programming]) for links with Global IPv6 addresses. The headend is required to resolve the specified Local IPv6 Address to the Node originating it and then use the Remote IPv6 Address to identify the link adjacency being referred to. The Local and Remote Address pair link descriptors follows semantics as specified in [RFC7752].

When the algorithm is not specified for the SID types above which optionally allow for it, the headend SHOULD use the Strict Shortest Path algorithm if available; otherwise it SHOULD use the default Shortest Path algorithm. The specification of algorithm enables the use of IGP Flex Algorithm [I-D.ietf-lsr-flex-algo] specific SIDs in SR Policy.

For SID types 3-through-11, a SID value may also be optionally provided to the headend for verification purposes. Section 5.1. describes the resolution and verification of the SIDs and Segment Lists on the headend.

When building the MPLS label stack or the IPv6 Segment list from the Segment List, the node instantiating the policy MUST interpret the set of Segments as follows:

- The first Segment represents the topmost label or the first IPv6 segment. It identifies the active segment the traffic will be directed toward along the explicit SR path.
- The last Segment represents the bottommost label or the last IPv6 segment the traffic will be directed toward along the explicit SR path.

4.1. Explicit Null

A Type 1 SID may be any MPLS label, including reserved labels.

For example, assuming that the desired traffic-engineered path from a headend 1 to an endpoint 4 can be expressed by the Segment-List <16002, 16003, 16004> where 16002, 16003 and 16004 respectively refer to the IPv4 Prefix SIDs bound to node 2, 3 and 4, then IPv6 traffic can be traffic-engineered from nodes 1 to 4 via the previously described path using an SR Policy with Segment-List <16002, 16003, 16004, 2> where mpls label value of 2 represents the "IPv6 Explicit NULL Label".

The penultimate node before node 4 will pop 16004 and will forward the frame on its directly connected interface to node 4.

The endpoint receives the traffic with top label "2" which indicates that the payload is an IPv6 packet.

When steering unlabeled IPv6 BGP destination traffic using an SR policy composed of Segment-List(s) based on IPv4 SIDs, the Explicit Null Label Policy is processed as specified in [I-D.ietf-idr-segment-routing-te-policy]) Section 2.4.4. When an "IPv6 Explicit NULL label" is not present as the bottom label, the headend SHOULD automatically impose one. Refer to Section 8 for more details.

5. Validity of a Candidate Path

5.1. Explicit Candidate Path

An explicit candidate path is associated with a Segment-List or a set of Segment-Lists.

An explicit candidate path is provisioned by the operator directly or via a controller.

The computation/logic that leads to the choice of the Segment-List is external to the SR Policy headend. The SR Policy headend does not compute the Segment-List. The SR Policy headend only confirms its validity.

A Segment-List of an explicit candidate path MUST be declared invalid when:

- It is empty.
- Its weight is 0.
- The headend is unable to perform path resolution for the first SID into one or more outgoing interface(s) and next-hop(s).
- The headend is unable to perform SID resolution for any non-first SID of type 3-through-11 into an MPLS label or an SRv6 SID.
- The headend verification fails for any SID for which verification has been explicitly requested.

"Unable to perform path resolution" means that the headend has no path to the SID in its SR database.

SID verification is performed when the headend is explicitly requested to verify SID(s) by the controller via the signaling protocol used. Implementations MAY provide a local configuration

option to enable verification on a global or per policy or per candidate path basis.

"Verification fails" for a SID means any of the following:

- The headend is unable to find the SID in its SR DB
- The headend detects mis-match between the SID value and its context provided for SIDs of type 3-through-11 in its SR DB.
- The headend is unable to perform SID resolution for any non-first SID of type 3-through-11 into an MPLS label or an SRv6 SID.

In multi-domain deployments, it is expected that the headend be unable to verify the reachability of the SIDs in remote domains. Types 1 or 2 MUST be used for the SIDs for which the reachability cannot be verified. Note that the first SID MUST always be reachable regardless of its type.

In addition, a Segment-List MAY be declared invalid when:

- Its last segment is not a Prefix SID (including BGP Peer Node-SID) advertised by the node specified as the endpoint of the corresponding SR policy.
- Its last segment is not an Adjacency SID (including BGP Peer Adjacency SID) of any of the links present on neighbor nodes and that terminate on the node specified as the endpoint of the corresponding SR policy.

An explicit candidate path is invalid as soon as it has no valid Segment-List.

5.2. Dynamic Candidate Path

A dynamic candidate path is specified as an optimization objective and constraints.

The headend of the policy leverages its SR database to compute a Segment-List ("solution Segment-List") that solves this optimization problem.

The headend re-computes the solution Segment-List any time the inputs to the problem change (e.g., topology changes).

When local computation is not possible (e.g., a policy's tailend is outside the topology known to the headend) or not desired, the headend MAY send path computation request to a PCE supporting PCEP extension specified in [I-D.ietf-pce-segment-routing].

If no solution is found to the optimization objective and constraints, then the dynamic candidate path MUST be declared invalid.

[I-D.filsfils-spring-sr-policy-considerations] discusses some of the optimization objectives and constraints that may be considered by a dynamic candidate path. It illustrates some of the desirable properties of the computation of the solution Segment-List.

6. Binding SID

The Binding SID (BSID) is fundamental to Segment Routing [RFC8402]. It provides scaling, network opacity and service independence. [I-D.filsfils-spring-sr-policy-considerations] illustrates some of these benefits. This section describes the association of BSID with an SR Policy.

6.1. BSID of a candidate path

Each candidate path MAY be defined with a BSID.

Candidate Paths of the same SR policy SHOULD have the same BSID.

Candidate Paths of different SR policies MUST NOT have the same BSID.

6.2. BSID of an SR Policy

The BSID of an SR Policy is the BSID of its active candidate path.

When the active candidate path has a specified BSID, the SR Policy uses that BSID if this value (label in MPLS, IPv6 address in SRv6) is available (i.e., not associated with any other usage: e.g. to another MPLS client, to another SID, to another SR Policy).

Optionally, instead of only checking that the BSID of the active path is available, a headend MAY check that it is available within a given SID range i.e., Segment Routing Local Block (SRLB) as specified in [RFC8402].

When the specified BSID is not available (optionally is not in the SRLB), an alert message MUST be generated.

In the cases (as described above) where SR Policy does not have a BSID available, then the SR Policy MAY dynamically bind a BSID to itself. Dynamically bound BSID SHOULD use an available SID outside the SRLB.

Assuming that at time t the BSID of the SR Policy is B1, if at time t+dt a different candidate path becomes active and this new active path does not have a specified BSID or its BSID is specified but is not available (e.g. it is in use by something else), then the SR Policy keeps the previous BSID B1.

The association of an SR Policy with a BSID thus MAY change over the life of the SR Policy (e.g., upon active path change). Hence, the BSID SHOULD NOT be used as an identification of an SR Policy.

6.2.1. Frequent use-case : unspecified BSID

All the candidate paths of the same SR Policy can have an unspecified BSID.

In such a case, a BSID MAY be dynamically bound to the SR Policy as soon as the first valid candidate path is received. That BSID is kept along all the life of the SR Policy and across changes of active candidate path.

6.2.2. Frequent use-case: all specified to the same BSID

All the paths of the SR Policy can have the same specified BSID.

6.2.3. Specified-BSID-only

An implementation MAY support the configuration of the Specified-BSID-only restrictive behavior on the headend for all SR Policies or individual SR Policies. Further, this restrictive behavior MAY also be signaled on a per SR Policy basis to the headend.

When this restrictive behavior is enabled, if the candidate path has an unspecified BSID or if the specified BSID is not available when the candidate path becomes active then no BSID is bound to it and it is considered invalid. An alert MUST be triggered to notify this error. Other candidate paths MUST then be evaluated for becoming the active candidate path.

6.3. Forwarding Plane

A valid SR Policy installs a BSID-keyed entry in the forwarding plane with the action of steering the packets matching this entry to the selected path of the SR Policy.

If the Specified-BSID-only restrictive behavior is enabled and the BSID of the active path is not available (optionally not in the SRLB), then the SR Policy does not install any entry indexed by a BSID in the forwarding plane.

6.4. Non-SR usage of Binding SID

An implementation MAY choose to associate a Binding SID with any type of interface (e.g. a layer 3 termination of an Optical Circuit) or a tunnel (e.g. IP tunnel, GRE tunnel, IP/UDP tunnel, MPLS RSVP-TE tunnel, etc). This enables the use of other non-SR enabled interfaces and tunnels as segments in an SR Policy Segment-List without the need of forming routing protocol adjacencies over them.

The details of this kind of usage are beyond the scope of this document. A specific packet optical integration use case is described in [I-D.anand-spring-poi-sr]

7. SR Policy State

The SR Policy State is maintained on the headend to represent the state of the policy and its candidate paths. This is to provide an accurate representation of whether the SR Policy is being instantiated in the forwarding plane and which of its candidate paths and segment-list(s) are active. The SR Policy state MUST also reflect the reason when a policy and/or its candidate path is not active due to validation errors or not being preferred.

The SR Policy state can be reported by the headend node via BGP-LS [I-D.ietf-idr-te-lsp-distribution] or PCEP [RFC8231] and [I-D.sivabalan-pce-binding-label-sid].

SR Policy state on the headend also includes traffic accounting information for the flows being steered via the policies. The details of the SR Policy accounting are beyond the scope of this document. The aspects related to the SR traffic counters and their usage in the broader context of traffic accounting in a SR network are covered in [I-D.filsfils-spring-sr-traffic-counters] and [I-D.ali-spring-sr-traffic-accounting] respectively.

Implementations MAY support an administrative state to control locally provisioned policies via mechanisms like CLI or NETCONF.

8. Steering into an SR Policy

A headend can steer a packet flow into a valid SR Policy in various ways:

- Incoming packets have an active SID matching a local BSID at the headend.
- Per-destination Steering: incoming packets match a BGP/Service route which recurses on an SR policy.

- Per-flow Steering: incoming packets match or recurse on a forwarding array of where some of the entries are SR Policies.
- Policy-based Steering: incoming packets match a routing policy which directs them on an SR policy.

For simplicity of illustration, this document uses the SR-MPLS example.

8.1. Validity of an SR Policy

An SR Policy is invalid when all its candidate paths are invalid as described in Section 5 and Section 2.10.

By default, upon transitioning to the invalid state,

- an SR Policy and its BSID are removed from the forwarding plane.
- any steering of a service (PW), destination (BGP-VPN), flow or packet on the related SR policy is disabled and the related service, destination, flow or packet is routed per the classic forwarding table (e.g. longest-match to the destination or the recursing next-hop).

8.2. Drop upon invalid SR Policy

An SR Policy MAY be enabled for the Drop-Upon-Invalid behavior:

- an invalid SR Policy and its BSID is kept in the forwarding plane with an action to drop.
- any steering of a service (PW), destination (BGP-VPN), flow or packet on the related SR policy is maintained with the action to drop all of this traffic.

The drop-upon-invalid behavior has been deployed in use-cases where the operator wants some PW to only be transported on a path with specific constraints. When these constraints are no longer met, the operator wants the PW traffic to be dropped. Specifically, the operator does not want the PW to be routed according to the IGP shortest-path to the PW endpoint.

8.3. Incoming Active SID is a BSID

Let us assume that headend H has a valid SR Policy P of Segment-List <S1, S2, S3> and BSID B.

When H receives a packet K with label stack <B, L2, L3>, H pops B and pushes <S1, S2, S3> and forwards the resulting packet according to SID S1.

"Forwarding the resulting packet according to S1" means: If S1 is an Adj SID or a PHP-enabled prefix SID advertised by a neighbor, H sends the resulting packet with label stack <S2, S3, L2, L3> on the outgoing interface associated with S1; Else H sends the resulting packet with label stack <S1, S2, S3, L2, L3> along the path of S1.

H has steered the packet into the SR policy P.

H did not have to classify the packet. The classification was done by a node upstream of H (e.g., the source of the packet or an intermediate ingress edge node of the SR domain) and the result of this classification was efficiently encoded in the packet header as a BSID.

This is another key benefit of the segment routing in general and the binding SID in particular: the ability to encode a classification and the resulting steering in the packet header to better scale and simplify intermediate aggregation nodes.

If the SR Policy P is invalid, the BSID B is not in the forwarding plane and hence the packet K is dropped by H.

8.4. Per-Destination Steering

Let us assume that headend H:

- learns a BGP route R/r via next-hop N, extended-color community C and VPN label V.
- has a valid SR Policy P to (color = C, endpoint = N) of Segment-List <S1, S2, S3> and BSID B.
- has a BGP policy which matches on the extended-color community C and allows its usage as SLA steering information.

If all these conditions are met, H installs R/r in RIB/FIB with next-hop = SR Policy P of BSID B instead of via N.

Indeed, H's local BGP policy and the received BGP route indicate that the headend should associate R/r with an SR Policy path to endpoint N with the SLA associated with color C. The headend therefore installs the BGP route on that policy.

This can be implemented by using the BSID as a generalized next-hop and installing the BGP route on that generalized next-hop.

When H receives a packet K with a destination matching R/r, H pushes the label stack <S1, S2, S3, V> and sends the resulting packet along the path to S1.

Note that any SID associated with the BGP route is inserted after the Segment-List of the SR Policy (i.e., <S1, S2, S3, V>).

The same behavior is applicable to any type of service route: any AFI/SAFI of BGP [RFC4760] any AFI/SAFI of LISP [RFC6830].

8.4.1. Multiple Colors

When a BGP route has multiple extended-color communities each with a valid SR Policy NLRI, the BGP process installs the route on the SR policy whose color is of highest numerical value.

Let us assume that headend H:

- learns a BGP route R/r via next-hop N, extended-color communities C1 and C2 and VPN label V.
- has a valid SR Policy P1 to (color = C1, endpoint = N) of Segment-List <S1, S2, S3> and BSID B1.
- has a valid SR Policy P2 to (color = C2, endpoint = N) of Segment-List <S4, S5, S6> and BSID B2.
- has a BGP policy which matches on the extended-color communities C1 and C2 and allows their usage as SLA steering information

If all these conditions are met, H installs R/r in RIB/FIB with next-hop = SR Policy P2 of BSID=B2 (instead of N) because C2 > C1.

8.5. Recursion on an on-demand dynamic BSID

In the previous section, it was assumed that H had a pre-established "explicit" SR Policy (color C, endpoint N).

In this section, independently to the a-priori existence of any explicit candidate path of the SR policy (C, N), it is to be noted that the BGP process at headend node H triggers the instantiation of a dynamic candidate path for the SR policy (C, N) as soon as:

- the BGP process learns of a route R/r via N and with color C.
- a local policy at node H authorizes the on-demand SR Policy path instantiation and maps the color to a dynamic SR Policy path optimization template.

8.5.1. Multiple Colors

When a BGP route R/r via N has multiple extended-color communities Ci (with i=1 ... n), an individual on-demand SR Policy dynamic path request (color Ci, endpoint N) is triggered for each color Ci.

8.6. Per-Flow Steering

Let us assume that headend H:

- has a valid SR Policy P1 to (color = C1, endpoint = N) of Segment-List <S1, S2, S3> and BSID B1.
- has a valid SR Policy P2 to (color = C2, endpoint = N) of Segment-List <S4, S5, S6> and BSID B2.
- is configured to instantiate an array of paths to N where the entry 0 is the IGP path to N, color C1 is the first entry and Color C2 is the second entry. The index into the array is called a Forwarding Class (FC). The index can have values 0 to 7.
- is configured to match flows in its ingress interfaces (upon any field such as Ethernet destination/source/vlan/tos or IP destination/source/DSCP or transport ports etc.) and color them with an internal per-packet forwarding-class variable (0, 1 or 2 in this example).

If all these conditions are met, H installs in RIB/FIB:

- N via a recursion on an array A (instead of the immediate outgoing link associated with the IGP shortest-path to N).
- Entry A(0) set to the immediate outgoing link of the IGP shortest-path to N.
- Entry A(1) set to SR Policy P1 of BSID=B1.
- Entry A(2) set to SR Policy P2 of BSID=B2.

H receives three packets K, K1 and K2 on its incoming interface. These three packets either longest-match on N or more likely on a BGP/service route which recurses on N. H colors these 3 packets respectively with forwarding-class 0, 1 and 2. As a result:

- H forwards K along the shortest-path to N (which in SR-MPLS results in the pushing of the prefix-SID of N).
- H pushes <S1, S2, S3> on packet K1 and forwards the resulting frame along the shortest-path to S1.
- H pushes <S4, S5, S6> on packet K2 and forwards the resulting frame along the shortest-path to S4.

If the local configuration does not specify any explicit forwarding information for an entry of the array, then this entry is filled with the same information as entry 0 (i.e. the IGP shortest-path).

If the SR Policy mapped to an entry of the array becomes invalid, then this entry is filled with the same information as entry 0. When all the array entries have the same information as entry0, the forwarding entry for N is updated to bypass the array and point directly to its outgoing interface and next-hop.

The array index values (e.g. 0, 1 and 2) and the notion of forwarding-class are implementation specific and only meant to describe the desired behavior. The same can be realized by other mechanisms.

This realizes per-flow steering: different flows bound to the same BGP endpoint are steered on different IGP or SR Policy paths.

A headend MAY support options to apply per-flow steering only for traffic matching specific prefixes (e.g. specific IGP or BGP prefixes).

8.7. Policy-based Routing

Finally, headend H may be configured with a local routing policy which overrides any BGP/IGP path and steer a specified packet on an SR Policy. This includes the use of mechanisms like IGP Shortcut for automatic routing of IGP prefixes over SR Policies intended for such purpose.

8.8. Optional Steering Modes for BGP Destinations

8.8.1. Color-Only BGP Destination Steering

In the previous section, it is seen that the steering on an SR Policy is governed by the matching of the BGP route's next-hop N and the authorized color C with an SR Policy defined by the tuple (N, C).

This is the most likely form of BGP destination steering and the one recommended for most use-cases.

This section defines an alternative steering mechanism based only on the color.

This color-only steering variation is governed by two new flags "C" and "O" defined in the color extended community [ref draft-ietf-idr-segment-routing-te-policy section 3].

The Color-Only flags "CO" are set to 00 by default.

When 00, the BGP destination is steered as follows:

```
    IF there is a valid SR Policy (N, C) where N is the IPv4 or IPv6

            endpoint address and C is a color;
        Steer into SR Policy (N, C);
    ELSE;
        Steer on the IGP path to the next-hop N.
```

This is the classic case described in this document previously and what is recommended in most scenarios.

When 01, the BGP destination is steered as follows:

```
    IF there is a valid SR Policy (N, C) where N is the IPv4 or IPv6

            endpoint address and C is a color;
        Steer into SR Policy (N, C);
    ELSE IF there is a valid SR Policy (null endpoint, C) of the
            same address-family of N;
        Steer into SR Policy (null endpoint, C);
    ELSE IF there is any valid SR Policy
            (any address-family null endpoint, C);
        Steer into SR Policy (any null endpoint, C);
    ELSE;
        Steer on the IGP path to the next-hop N.
```

When 10, the BGP destination is steered as follows:

```
    IF there is a valid SR Policy (N, C) where N is an IPv4 or IPv6
            endpoint address and C is a color;
        Steer into SR Policy (N, C);
    ELSE IF there is a valid SR Policy (null endpoint, C)
            of the same address-family of N;
        Steer into SR Policy (null endpoint, C);
    ELSE IF there is any valid SR Policy
            (any address-family null endpoint, C);
        Steer into SR Policy (any null endpoint, C);
    ELSE IF there is any valid SR Policy (any endpoint, C)
            of the same address-family of N;
        Steer into SR Policy (any endpoint, C);
    ELSE IF there is any valid SR Policy
            (any address-family endpoint, C);
        Steer into SR Policy (any address-family endpoint, C);
    ELSE;
        Steer on the IGP path to the next-hop N.
```

The null endpoint is 0.0.0.0 for IPv4 and ::0 for IPv6 (all bits set to the 0 value).

The value 11 is reserved for future use and SHOULD NOT be used. Upon reception, an implementations MUST treat it like 00.

8.8.2. Multiple Colors and CO flags

The steering preference is first based on highest color value and then CO-dependent for the color. Assuming a Prefix via (NH, C1(CO=01), C2(CO=01)); C1>C2 The steering preference order is:

- SR policy (NH, C1).
- SR policy (null, C1).
- SR policy (NH, C2).
- SR policy (null, C2).
- IGP to NH.

8.8.3. Drop upon Invalid

This document defined earlier that when all the following conditions are met, H installs R/r in RIB/FIB with next-hop = SR Policy P of BSID B instead of via N.

- H learns a BGP route R/r via next-hop N, extended-color community C and VPN label V.
- H has a valid SR Policy P to (color = C, endpoint = N) of Segment-List <S1, S2, S3> and BSID B.
- H has a BGP policy which matches on the extended-color community C and allows its usage as SLA steering information.

This behavior is extended by noting that the BGP policy may require the BGP steering to always stay on the SR policy whatever its validity.

This is the "drop upon invalid" option described in Section 8.2 applied to BGP-based steering.

9. Protection

9.1. Leveraging TI-LFA local protection of the constituent IGP segments

In any topology, Topology-Independent Loop Free Alternate (TI-LFA) [I-D.bashandy-rtgwg-segment-routing-ti-lfa] provides a 50msec local protection technique for IGP SIDs. The backup path is computed on a per IGP SID basis along the post-convergence path.

In a network that has deployed TI-LFA, an SR Policy built on the basis of TI-LFA protected IGP segments leverages the local protection of the constituent segments.

In a network that has deployed TI-LFA, an SR Policy instantiated only with non-protected Adj SIDs does not benefit from any local protection.

9.2. Using an SR Policy to locally protect a link

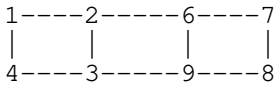


Figure 1: Local protection using SR Policy

An SR Policy can be instantiated at node 2 to protect the link 2to6. A typical explicit Segment-List would be <3, 9, 6>.

A typical use-case occurs for links outside an IGP domain: e.g. 1, 2, 3 and 4 are part of IGP/SR sub-domain 1 while 6, 7, 8 and 9 are part of IGP/SR sub-domain 2. In such a case, links 2to6 and 3to9 cannot benefit from TI-LFA automated local protection. The SR Policy with Segment-List <3, 9, 6> on node 2 can be locally configured to be a fast-reroute backup path for the link 2to6.

9.3. Using a Candidate Path for Path Protection

An SR Policy allows for multiple candidate paths, of which at any point in time there is a single active candidate path that is provisioned in the forwarding plane and used for traffic steering. However, another (lower preference) candidate path MAY be designated as the backup for a specific or all (active) candidate path(s). Following options are possible:

- A pair of disjoint candidate paths are provisioned with one of them as primary and the other is identified as its backup.
- A specific candidate path is provisioned as the backup for any (active) candidate path.
- The headend picks the next (lower) preference valid candidate path as the backup for the active candidate path.

The headend MAY compute a-priori and validate such backup candidate paths as well as provision them into forwarding plane as backup for the active path. A fast re-route mechanism MAY then be used to trigger sub 50msec switchover from the active to the backup candidate path in the forwarding plane. Mechanisms like BFD MAY be used for fast detection of such failures.

10. Security Considerations

This document does not define any new protocol extensions and does not impose any additional security challenges.

11. IANA Considerations

This document has no actions for IANA.

12. Acknowledgement

The authors would like to thank Tarek Saad, Dhanendra Jain, Ruediger Geib and Rob Shakir for their valuable comments and suggestions.

13. Contributors

The following people have contributed to this document:

Ketan Talaulikar
Cisco Systems
Email: ketant@cisco.com

Zafar Ali
Cisco Systems
Email: zali@cisco.com

Jose Liste
Cisco Systems
Email: jliste@cisco.com

Francois Clad
Cisco Systems
Email: fclad@cisco.com

Kamran Raza
Cisco Systems
Email: skraza@cisco.com

Shraddha Hegde
Juniper Networks
Email: shraddha@juniper.net

Steven Lin
Google, Inc.
Email: stevenlin@google.com

Przemyslaw Krol
Google, Inc.
Email: pkrol@google.com

Martin Horneffer
Deutsche Telekom
Email: martin.horneffer@telekom.de

Dirk Steinberg
Steinberg Consulting
Email: dws@steinbergnet.net

Bruno Decraene
Orange Business Services
Email: bruno.decraene@orange.com

Stephane Litkowski
Orange Business Services
Email: stephane.litkowski@orange.com

Luay Jalil
Verizon
Email: luay.jalil@verizon.com

14. References

14.1. Normative References

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

[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>.

14.2. Informative References

[I-D.ali-spring-sr-traffic-accounting] Ali, Z., Filsfils, C., Talaulikar, K., Sivabalan, S., Horneffer, M., Raszuk, R., Litkowski, S., and d. daniel.voyer@bell.ca, "Traffic Accounting in Segment Routing Networks", draft-ali-spring-sr-traffic-accounting-02 (work in progress), June 2018.

[I-D.anand-spring-poi-sr] Anand, M., Bardhan, S., Subrahmaniam, R., Tantsura, J., Mukhopadhyaya, U., and C. Filsfils, "Packet-Optical Integration in Segment Routing", draft-anand-spring-poi-sr-06 (work in progress), July 2018.

[I-D.bashandy-rtgwg-segment-routing-ti-lfa]
Bashandy, A., Filsfils, C., Decraene, B., Litkowski, S., Francois, P., daniel.voyer@bell.ca, d., Clad, F., and P. Camarillo, "Topology Independent Fast Reroute using Segment Routing", draft-bashandy-rtgwg-segment-routing-ti-lfa-05 (work in progress), October 2018.

[I-D.filsfils-spring-sr-policy-considerations]
Filsfils, C., Talaulikar, K., Krol, P., Horneffer, M., and P. Mattes, "SR Policy Implementation and Deployment Considerations", draft-filsfils-spring-sr-policy-considerations-02 (work in progress), October 2018.

[I-D.filsfils-spring-sr-traffic-counters]
Filsfils, C., Ali, Z., Horneffer, M., daniel.voyer@bell.ca, d., Durrani, M., and R. Raszuk, "Segment Routing Traffic Accounting Counters", draft-filsfils-spring-sr-traffic-counters-00 (work in progress), June 2018.

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-05 (work in progress), July 2018.

[I-D.ietf-idr-bgp-ls-segment-routing-msd]
Tantsura, J., Chunduri, U., Mirsky, G., and S. Sivabalan, "Signaling MSD (Maximum SID Depth) using Border Gateway Protocol Link-State", draft-ietf-idr-bgp-ls-segment-routing-msd-02 (work in progress), August 2018.

[I-D.ietf-idr-bgpls-segment-routing-epe]
Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", draft-ietf-idr-bgpls-segment-routing-epe-17 (work in progress), October 2018.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Jain, D., Mattes, P., Rosen, E., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-04 (work in progress), July 2018.

[I-D.ietf-idr-te-lsp-distribution]
Previdi, S., Talaulikar, K., Dong, J., Chen, M., Gredler, H., and J. Tantsura, "Distribution of Traffic Engineering (TE) Policies and State using BGP-LS", draft-ietf-idr-te-lsp-distribution-09 (work in progress), June 2018.

[I-D.ietf-isis-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling MSD (Maximum SID Depth) using IS-IS", draft-ietf-isis-segment-routing-msd-19 (work in progress), October 2018.

[I-D.ietf-lsr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-algo-00 (work in progress), May 2018.

[I-D.ietf-ospf-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling MSD (Maximum SID Depth) using OSPF", draft-ietf-ospf-segment-routing-msd-25 (work in progress), October 2018.

[I-D.ietf-pce-segment-routing]
Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "PCEP Extensions for Segment Routing", draft-ietf-pce-segment-routing-14 (work in progress), October 2018.

[I-D.sivabalan-pce-binding-label-sid]
Sivabalan, S., Filsfils, C., Tantsura, J., Hardwick, J., Previdi, S., and D. Dhody, "Carrying Binding Label/ Segment-ID in PCE-based Networks.", draft-sivabalan-pce-binding-label-sid-05 (work in progress), October 2018.

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <https://www.rfc-editor.org/info/rfc1195>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <https://www.rfc-editor.org/info/rfc3630>.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <https://www.rfc-editor.org/info/rfc4456>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <https://www.rfc-editor.org/info/rfc4760>.

[RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <https://www.rfc-editor.org/info/rfc5305>.

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

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

[RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <https://www.rfc-editor.org/info/rfc7471>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC7810] Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 7810, DOI 10.17487/RFC7810, May 2016, <https://www.rfc-editor.org/info/rfc7810>.

[RFC8231] Crabbe, E., Minei, I., Medved, J., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE", RFC 8231, DOI 10.17487/RFC8231, September 2017, <https://www.rfc-editor.org/info/rfc8231>.

[RFC8281] Crabbe, E., Minei, I., Sivabalan, S., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for PCE-Initiated LSP Setup in a Stateful PCE Model", RFC 8281, DOI 10.17487/RFC8281, December 2017, <https://www.rfc-editor.org/info/rfc8281>.

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Pegasus Parc
De kleetlaan 6a, DIEGEM BRABANT 1831
BELGIUM

Email: cfilsfil@cisco.com

Siva Sivabalan (editor)
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, Ontario K2K 3E8
Canada

Email: msiva@cisco.com

Daniel Voyer
Bell Canada
671 de la gauchetiere W
Montreal, Quebec H3B 2M8
Canada

Email: daniel.voyer@bell.ca

Alex Bogdanov
Google, Inc.

Email: bogdanov@google.com

Paul Mattes
Microsoft
One Microsoft Way
Redmond, WA 98052-6399
USA

Email: pamattes@microsoft.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2018

S. Litkowski
Orange Business Service
Y. Qu
Huawei
P. Sarkar
Individual
J. Tantsura
Nuage Networks

June 28, 2018

YANG Data Model for Segment Routing
draft-ietf-spring-sr-yang-09

Abstract

This document defines a YANG data model ([RFC6020], [RFC7950]) for segment routing ([I-D.ietf-spring-segment-routing]) configuration and operation. This YANG model is intended to be used on network elements to configure or operate segment routing. This document defines also generic containers that SHOULD be reused by IGP protocol modules to support segment routing.

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 December 30, 2018.

Table of Contents

1. Introduction

This document defines a YANG data model for segment routing configuration and operation. This document does not define the IGP extensions to support segment routing but defines generic groupings that SHOULD be reused by IGP extension modules. The reason of this design choice is to not require implementations to support all IGP extensions. For example, an implementation may support IS-IS extension but not OSPF.

1.1. Tree diagram

Tree diagrams used in this document follow the notation defined in [I-D.ietf-netmod-yang-tree-diagrams].

2. Design of the Data Model

As the module definition is just starting, it is expected that there will be changes as the module matures.

```
module: ietf-segment-routing
  augment /rt:routing:
    +--rw segment-routing
       +--rw transport-type?      identityref
       +--ro node-capabilities
       |  +--ro transport-planes* [transport-plane]
       |  |  +--ro transport-plane    identityref
       |  +--ro entropy-readable-label-depth?   uint8
       +--rw msd {msd}?
       |  +--rw node-msd?   uint8
       |  +--rw link-msd
       |     +--rw link-msds* [interface]
       |        +--rw interface    if:interface-ref
       |        +--rw msd?         uint8
       +--rw bindings
       |  +--rw mapping-server {mapping-server}?
       |  |  +--rw policy* [name]
       |  |     +--rw name    string
       |  |     +--rw ipv4
       |  |     |  +--rw mapping-entry* [prefix algorithm]
       |  |     |     +--rw prefix        inet:ipv4-prefix
       |  |     |     +--rw value-type?   enumeration
       |  |     |     +--rw start-sid     uint32
       |  |     |     +--rw range?        uint32
       |  |     |     +--rw algorithm     identityref
       |  |     +--rw ipv6
       |  |        +--rw mapping-entry* [prefix algorithm]
       |  |           +--rw prefix        inet:ipv6-prefix
       |  |           +--rw value-type?   enumeration
       |  |           +--rw start-sid     uint32
       |  |           +--rw range?        uint32
       |  |           +--rw algorithm     identityref
       |  +--rw connected-prefix-sid-map
       |  |  +--rw ipv4
       |  |  |  +--rw ipv4-prefix-sid* [prefix algorithm]
       |  |  |     +--rw prefix               inet:ipv4-prefix
       |  |  |     +--rw value-type?          enumeration
       |  |  |     +--rw start-sid            uint32
```

```
       |  |  |     +--rw range?               uint32
       |  |  |     +--rw algorithm            identityref
       |  |  |     +--rw last-hop-behavior?   enumeration {sid-last-hop-behavior
}?
       |  |  +--rw ipv6
       |  |     +--rw ipv6-prefix-sid* [prefix algorithm]
       |  |        +--rw prefix               inet:ipv6-prefix
       |  |        +--rw value-type?          enumeration
       |  |        +--rw start-sid            uint32
       |  |        +--rw range?               uint32
       |  |        +--rw algorithm            identityref
       |  |        +--rw last-hop-behavior?   enumeration {sid-last-hop-behavior
}?
       |  +--rw local-prefix-sid
       |     +--rw ipv4
       |     |  +--rw ipv4-prefix-sid-local* [prefix algorithm]
       |     |     +--rw prefix        inet:ipv4-prefix
       |     |     +--rw value-type?   enumeration
       |     |     +--rw start-sid     uint32
       |     |     +--rw range?        uint32
       |     |     +--rw algorithm     identityref
       |     +--rw ipv6
       |        +--rw ipv6-prefix-sid-local* [prefix algorithm]
       |           +--rw prefix        inet:ipv6-prefix
       |           +--rw value-type?   enumeration
       |           +--rw start-sid     uint32
       |           +--rw range?        uint32
       |           +--rw algorithm     identityref
       +--rw global-srgb
       |  +--rw srgb* [lower-bound upper-bound]
       |     +--rw lower-bound    uint32
       |     +--rw upper-bound    uint32
       +--rw srlb
       |  +--rw srlb* [lower-bound upper-bound]
       |     +--rw lower-bound    uint32
       |     +--rw upper-bound    uint32
       +--ro label-blocks*
       |  +--ro lower-bound?   uint32
       |  +--ro upper-bound?   uint32
       |  +--ro size?          uint32
       |  +--ro free?          uint32
       |  +--ro used?          uint32
       |  +--ro scope?         enumeration
       +--ro sid-list
          +--ro sid* [target sid source source-protocol binding-type]
             +--ro target             string
             +--ro sid                uint32
             +--ro algorithm?         uint8
             +--ro source             inet:ip-address
             +--ro used?              boolean
```

```
            +--ro source-protocol    -> /rt:routing/control-plane-protocols
            +                           /control-plane-protocol/name
            +--ro binding-type       enumeration
            +--ro scope?             enumeration

  notifications:
    +---n segment-routing-global-srgb-collision
    |  +--ro srgb-collisions*
    |     +--ro lower-bound?          uint32
    |     +--ro upper-bound?          uint32
    |     +--ro routing-protocol?     -> /rt:routing/control-plane-protocols
    |                                    /control-plane-protocol/name
    |     +--ro originating-rtr-id?   router-id
    +---n segment-routing-global-sid-collision
    |  +--ro received-target?       string
    |  +--ro new-sid-rtr-id?        router-id
    |  +--ro original-target?       string
    |  +--ro original-sid-rtr-id?   router-id
    |  +--ro index?                 uint32
    |  +--ro routing-protocol?      -> /rt:routing/control-plane-protocols
    |                                  /control-plane-protocol/name
    +---n segment-routing-index-out-of-range
       +--ro received-target?    string
       +--ro received-index?     uint32
       +--ro routing-protocol?   -> /rt:routing/control-plane-protocols
                                    /control-plane-protocol/name
```

3. Configuration

This module augments the "/rt:routing:" with a segment-routing container. This container defines all the configuration parameters related to segment-routing.

The segment-routing configuration is split in global configuration and interface configuration.

The global configuration includes :

- segment-routing transport type : The underlying transport type for segment routing. The version of the model limits the transport type to an MPLS dataplane. The transport-type is only defined once for a particular routing-instance and is agnostic to the control plane used. Only a single transport-type is supported in this version of the model.

- bindings : Defines prefix to SID mappings. The operator can control advertisement of Prefix-SID independently for IPv4 and IPv6. Two types of mappings are available :

  * Mapping-server : maps non local prefixes to a segment ID. Configuration of bindings does not automatically allow advertisement of those bindings. Advertisement must be controlled by each routing-protocol instance (see Section 4). Multiple mapping policies may be defined.

  * Connected prefixes : maps connected prefixes to a segment ID. Advertisement of the mapping will be done by IGP when enabled for segment routing (see Section 4). The SID value can be expressed as an index (default), or an absolute value. The "last-hop-behavior" configuration dictates the PHP behavior: "explicit-null", "php", or "non-php".

- SRGB (Segment Routing Global Block): Defines a list of label blocks represented by a pair of lower-bound/upper-bound labels. The SRGB is also agnostic to the control plane used. So all routing-protocol instance will have to advertise the same SRGB.

- SRLB (Segment Routing Local Block): Defines a list of label blocks represented by a pair of lower-bound/upper-bound labels, reserved for lcoal SIDs.

4. IGP Control plane configuration

Support of segment-routing extensions for a particular IGP control plane is done by augmenting routing-protocol configuration with segment-routing extensions. This augmentation SHOULD be part of separate YANG modules in order to not create any dependency for implementations to support all protocol extensions.

This module defines groupings that SHOULD be used by IGP segment routing modules.

The "controlplane-cfg" grouping defines the generic global configuration for the IGP.

The "enabled" leaf enables segment-routing extensions for the routing-protocol instance.

The "bindings" container controls the routing-protocol instance’s advertisement of local bindings and the processing of received bindings.

4.1. IGP interface configuration

The interface configuration is part of the "igp-interface-cfg" grouping and includes Adjacency SID properties.

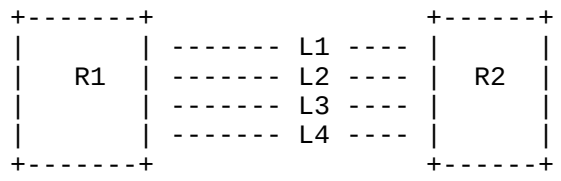
4.1.1. Adjacency SID properties

4.1.1.1. Bundling

This section is a first proposal on how to use S-bit in Adj-SID to create bundles. Authors would like to trigger discussion based on this first proposal.

In case of parallel IP links between routers, an additional Adjacency SID may be advertised representing more than one adjacency (i.e., a bundle of adjacencies). The "advertise-adj-group-sid" configuration controls whether or not an additional adjacency SID is advertised.

The "advertise-adj-group-sid" would be a list of "group-id". The "group-id" will permit to identify interfaces that must be bundled together.



In the figure above, R1 and R2 are interconnected by four links. A routing protocol adjacency is established on each link. Operator would like to create segment-routing Adj-SID that represent some bundles of links. We can imagine two different bundles : L1/L2 and L2/L3. To achieve this behavior, the service provider will configure a "group-id" X for both interfaces L1 and L2 and a "group-id" Y for both interfaces L3 and L3. This will result in R1 advertising an additional Adj-SID for each adjacency, for example a Adj-SID with S flag set and value of 400 will be added to L1 and L2. A Adj-SID with S flag set and value of 500 will be added to L3 and L4. As L1/L2 and L3/L4 does not share the same "group-id", a different SID value will be allocated.

4.1.1.2. Protection

The "advertise-protection" defines how protection for an interface is advertised. It does not control the activation or deactivation of protection. If the "single" option is used, a single Adj-SID will be advertised for the interface. If the interface is protected, the B-Flag for the Adj-SID advertisement will be set. If the "dual" option is used and if the interface is protected, two Adj-SIDs will be advertised for the interface adjacencies. One Adj-SID will always have the B-Flag set and the other will have the B-Flag clear. This

option is intended to be used in the case of traffic engineering where a path must use either protected segments or non-protected segments.

5. States

The operational states contains information reflecting the usage of allocated SRGB labels.

It also includes a list of all global SIDs, their associated bindings, and other information such as the source protocol and algorithm.

6. Notifications

The model defines the following notifications for segment-routing.

- segment-routing-global-srgb-collision: Rasied when a control plan advertised SRGB blocks have conflicts.
- segment-routing-global-sid-collision: Raised when a control plane advertised index is already associated with another target (in this version, the only defined targets are IPv4 and IPv6 prefixes).
- segment-routing-index-out-of-range: Raised when a control plane advertised index fall outside the range of SRGBs configured for the network device.

7. YANG Module

```
<CODE BEGINS> file "ietf-segment-routing-common@2018-06-25.yang"
module ietf-segment-routing-common {
  namespace "urn:ietf:params:xml:ns:yang:ietf-segment-routing-common";
  prefix sr-cmn;

  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF SPRING - SPRING Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/spring/>
     WG List:  <mailto:spring@ietf.org>

     Editor:    Stephane Litkowski
```

```
               <mailto:stephane.litkowski@orange.com>
     Editor:    Yingzhen Qu
               <mailto:yingzhen.qu@huawei.com>

     Author:    Acee Lindem
               <mailto:acee@cisco.com>
     Author:    Pushpasis Sarkar
               <mailto:pushpasis.ietf@gmail.com>
     Author:    Jeff Tantsura
               <jefftant.ietf@gmail.com>

    ";
  description
    "The YANG module defines a collection of types and groupings for
     Segment routing.

     Copyright (c) 2017 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX;
     see the RFC itself for full legal notices.";

  reference "RFC XXXX";

  revision 2018-06-25 {
    description
      "
       * Renamed readable-label-stack-depth to entropy-readable-label-depth;
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2017-07-01 {
    description
      "
        *Conform to RFC6087BIS Appendix C
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2017-03-10 {
    description
      "
```

```
       * Add support of SRLB
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-10-28 {
    description
      "
       * Add support of MSD (Maximum SID Depth)
       * Update contact info
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-10-24 {
    description
      "Initial";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }

  feature sid-last-hop-behavior {
    description
      "Configurable last hop behavior.";
  }

  identity segment-routing-transport {
    description
      "Base identity for segment routing transport.";
  }

  identity segment-routing-transport-mpls {
    base segment-routing-transport;
    description
      "This identity represents MPLS transport for segment
       routing.";
  }

  identity segment-routing-transport-ipv6 {
    base segment-routing-transport;
    description
      "This identity represents IPv6 transport for segment
       routing.";
  }

  identity prefix-sid-algorithm {
    description
      "Base identity for prefix-sid algorithm.";
  }

  identity prefix-sid-algorithm-shortest-path {
```

```
    base prefix-sid-algorithm;
    description
      "The default behavior of prefix-sid algorithm.";
  }

  identity prefix-sid-algorithm-strict-spf {
    base prefix-sid-algorithm;
    description
      "This algorithm mandates that the packet is forwared
       according to ECMP-aware SPF algorithm.";
  }

  grouping srlr {
    description
      "Grouping for SR Label Range configuration.";
    leaf lower-bound {
        type uint32;
        description
          "Lower value in the block.";
      }
    leaf upper-bound {
        type uint32;
        description
          "Upper value in the block.";
      }
  }

  grouping srgb-cfg {
    description
      "Grouping for SR Label Range configuration.";
    list srgb {
      key "lower-bound upper-bound";
      ordered-by user;
      description
        "List of global blocks to be
         advertised.";
      uses srlr;
    }
  }

  grouping srlb-cfg {
    description
      "Grouping for SR Local Block range configuration.";
    list srlb {
      key "lower-bound upper-bound";
      ordered-by user;
      description
        "List of SRLBs.";
```

```
      uses srlr;
    }
  }

  grouping sid-value-type {
    description
      "Defines how the SID value is expressed.";
    leaf value-type {
      type enumeration {
        enum "index" {
          description
            "The value will be
             interpreted as an index.";
        }
        enum "absolute" {
          description
            "The value will become
             interpreted as an absolute
             value.";
        }
      }
      default "index";
      description
        "This leaf defines how value
         must be interpreted.";
    }
  }

  grouping ipv4-sid-cfg {
    description
      "This grouping defines cfg of prefix SID.";
    leaf prefix {
      type inet:ipv4-prefix;
      description
        "connected prefix sid.";
    }
    uses prefix-sid-attributes;
  }
  grouping ipv6-sid-cfg {
    description
      "This grouping defines cfg of prefix SID.";
    leaf prefix {
      type inet:ipv6-prefix;
      description
        "connected prefix sid.";
    }
    uses prefix-sid-attributes;
  }
```

```
  grouping last-hop-behavior {
    description
      "Defines last hop behavior";
    leaf last-hop-behavior {
      if-feature "sid-last-hop-behavior";
      type enumeration {
        enum "explicit-null" {
          description
            "Use explicit-null for the SID.";
        }
        enum "no-php" {
          description
            "Do no use PHP for the SID.";
        }
        enum "php" {
          description
            "Use PHP for the SID.";
        }
      }
      description
        "Configure last hop behavior.";
    }
  }

  grouping node-capabilities {
    description
      "Containing SR node capabilities.";
    container node-capabilities {
      config false;
      description
        "Shows the SR capability of the node.";
      list transport-planes {
        key "transport-plane";
        description
          "List of supported transport planes.";
        leaf transport-plane {
          type identityref {
            base segment-routing-transport;
          }
          description
            "Transport plane supported";
        }
      }
      leaf entropy-readable-label-depth {
        type uint8;
        description
          "Maximum label statck depth that
           the router can read. ";
```

```
      }
    }
  }

  grouping prefix-sid-attributes {
    description
      "Containing SR attributes for a prefix.";
    uses sid-value-type;
    leaf start-sid {
      type uint32;
      mandatory true;
      description
        "Value associated with
         prefix. The value must
         be interpreted in the
         context of value-type.";
    }
    leaf range {
      type uint32;
      description
        "Describes how many SIDs could be
         allocated.";
    }
    leaf algorithm {
      type identityref {
        base prefix-sid-algorithm;
      }
      description
        "Prefix-sid algorithm.";
    }
  }
}
<CODE ENDS>
<CODE BEGINS> file "ietf-segment-routing@2018-06-25.yang"
module ietf-segment-routing {
  namespace "urn:ietf:params:xml:ns:yang:ietf-segment-routing";
  prefix sr;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-routing {
    prefix rt;
  }
  import ietf-interfaces {
```

```
    prefix if;
  }
  import ietf-routing-types {
    prefix rt-types;
  }
  import ietf-segment-routing-common {
    prefix sr-cmn;
  }

  organization
    "IETF SPRING - SPRING Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/spring/>
     WG List:  <mailto:spring@ietf.org>

     Editor:    Stephane Litkowski
                <mailto:stephane.litkowski@orange.com>
     Editor:    Yingzhen Qu
                <mailto:yingzhen.qu@huawei.com>

     Author:    Acee Lindem
                <mailto:acee@cisco.com>
     Author:    Pushpasis Sarkar
                <mailto:pushpasis.ietf@gmail.com>
     Author:    Jeff Tantsura
                <jefftant.ietf@gmail.com>

    ";
  description
    "The YANG module defines a generic configuration model for
     Segment routing common across all of the vendor
     implementations.

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

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX;
     see the RFC itself for full legal notices.";

  reference "RFC XXXX";
```

```
  revision 2018-06-25 {
    description
      "";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2017-07-01 {
    description
      "
       * Implement NMDA model
       *Conform to RFC6087BIS Appendix C
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }

  revision 2017-03-10 {
    description
      "
       * Change global-sid-list to sid-list and add a leaf scope
       * Added support of SRLB
       * Added support of local sids
       * fixed indentations
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-10-28 {
    description
      "
       * Add support of MSD (Maximum SID Depth)
       * Update contact info
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-10-24 {
    description
      "
       * Moved common SR types and groupings to a seperate module
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-07-07 {
    description
      "
       * Add support of prefix-sid algorithm configuration
       * change routing-protocols to control-plane-protocols
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-03-17 {
```

```
    description
      "
       * Add notification segment-routing-global-srgb-collision
       * Add router-id to segment-routing-global-sid-collision
       * Remove routing-instance
       * Add typedef router-id
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2015-10-17 {
    description
      "
       * Add per-protocol SRGB config feature
       * Move SRBG config to a grouping
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2015-06-22 {
    description
      "
       * Prefix SID config moved to
       connected-prefix-sid-map in global SR cfg
       rather than IGP.
      ";
    reference "draft-litkowski-spring-sr-yang-01";
  }
  revision 2015-04-23 {
    description
      "
       * Node flag deprecated from prefixSID
       * SR interface cfg moved to protocol
       * Adding multiple binding policies for SRMS
      ";
    reference "";
  }
  revision 2015-02-27 {
    description
      "Initial";
    reference "draft-litkowski-spring-sr-yang-00";
  }

  feature mapping-server {
    description
      "Support of SRMS.";
  }

  feature protocol-srgb {
    description
```

```
      "Support per-protocol srgb configuration.";
  }

  feature msd {
    description
      "Support of signaling MSD (Maximum SID Depth) in IGP.";
  }

  typedef system-id {
    type string {
      pattern "[0-9A-Fa-f]{4}\\.[0-9A-Fa-f]{4}\\.[0-9A-Fa-f]{4}\\.00";
    }
    description
      "This type defines ISIS system id using pattern,
       system id looks like : 0143.0438.AeF0.00";
  }

  typedef router-id {
    type union {
      type system-id;
      type rt-types:router-id;
    }
    description
      "OSPF/BGP router id or ISIS system ID.";
  }

  grouping controlplane-cfg {
    description
      "Defines protocol configuration.";
    container segment-routing {
      description
        "segment routing global config.";
      leaf enabled {
        type boolean;
        default "false";
        description
          "Enables segment-routing
           protocol extensions.";
      }
      container bindings {
        description
          "Control of binding advertisement
           and reception.";
        container advertise {
          description
            "Authorize the advertise
             of local mappings in binding TLV.";
          leaf-list policies {
```

```
            type string;
            description
              "List of policies to be advertised.";
          }
        }
        leaf receive {
          type boolean;
          default "true";
          description
            "Authorize the reception and usage
             of binding TLV.";
        }
      }
    }
  }

  grouping igp-interface-cfg {
    description
      "Grouping for IGP interface cfg.";
    container segment-routing {
      description
        "container for SR interface cfg.";
      container adjacency-sid {
        description
          "Defines the adjacency SID properties.";
        list advertise-adj-group-sid {
          key "group-id";
          description
            "Control advertisement of S flag.
             Enable to advertise a common Adj-SID
             for parallel links.";
          leaf group-id {
            type uint32;
            description
              "The value is an internal value to identify
               a group-ID. Interfaces with the same
               group-ID will be bundled together.";
          }
        }
        leaf advertise-protection {
          type enumeration {
            enum "single" {
              description
                "A single Adj-SID is associated
                 with the adjacency and reflects
                 the protection configuration.";
            }
            enum "dual" {
```

```
              description
                "Two Adj-SIDs will be associated
                 with the adjacency if interface
                 is protected. In this case
                 one will be enforced with
                 backup flag set, the other
                 will be enforced to backup flag unset.
                 In case, protection is not configured,
                 a single Adj-SID will be advertised
                 with backup flag unset.";
            }
          }
          description
            "If set, the Adj-SID refers to an
             adjacency being protected.";
        }
      }
    }
  }

  grouping msd-cfg {
    description
      "MSD configuration grouping.";
    leaf node-msd {
      type uint8;
      description
        "Node MSD is the lowest MSD supported by the node.";
    }
    container link-msd {
      description
        "Link MSD is a number represetns the particular link MSD value.";
      list link-msds {
        key "interface";
        description
          "List of link MSDs.";
        leaf interface {
          type if:interface-ref;
          description
            "Name of the interface.";
        }
        leaf msd {
          type uint8;
          description
            "SID depth of the interface associated with the link.";
        }
      }
    }
  }
```

```
  augment "/rt:routing" {
    description
      "This augments routing data model (RFC 8349)
       with segment-routing.";
    container segment-routing {
      description
        "segment routing global config.";
      leaf transport-type {
        type identityref {
          base sr-cmn:segment-routing-transport;
        }
        default "sr-cmn:segment-routing-transport-mpls";
        description
          "Dataplane to be used.";
      }
      uses sr-cmn:node-capabilities;
      container msd {
        if-feature "msd";
        description
          "MSD configuration.";
        uses msd-cfg;
      }
      container bindings {
        description
          "List of bindings.";
        container mapping-server {
          if-feature "mapping-server";
          description
            "Configuration of mapping-server
             local entries.";
          list policy {
            key "name";
            description
              "Definition of mapping policy.";
            leaf name {
              type string;
              description
                "Name of the mapping policy.";
            }
            container ipv4 {
              description
                "IPv4 mapping entries.";
              list mapping-entry {
                key "prefix algorithm";
                description
                  "Mapping entries.";
                uses sr-cmn:ipv4-sid-cfg;
              }
```

```
            }
            container ipv6 {
              description
                "IPv6 mapping entries.";
              list mapping-entry {
                key "prefix algorithm";
                description
                  "Mapping entries.";
                uses sr-cmn:ipv6-sid-cfg;
              }
            }
          }
        }
        container connected-prefix-sid-map {
          description
            "Prefix SID configuration.";
          container ipv4 {
            description
              "Parameters associated with IPv4 prefix SID";
            list ipv4-prefix-sid {
              key "prefix algorithm";
              description
                "List of prefix SID
                 mapped to IPv4 local prefixes.";
              uses sr-cmn:ipv4-sid-cfg;
              uses sr-cmn:last-hop-behavior;
            }
          }
          container ipv6 {
            description
              "Parameters associated with IPv6 prefix SID";
            list ipv6-prefix-sid {
              key "prefix algorithm";
              description
                "List of prefix SID
                 mapped to IPv6 local prefixes.";
              uses sr-cmn:ipv6-sid-cfg;
              uses sr-cmn:last-hop-behavior;
            }
          }
        }
        container local-prefix-sid {
          description
            "Local sid configuration.";
          container ipv4 {
            description
              "List of local ipv4 sids.";
            list ipv4-prefix-sid-local {
```

```
            key "prefix algorithm";
            description
              "List of local prefix-sid.";
            uses sr-cmn:ipv4-sid-cfg;
          }
        }
        container ipv6 {
          description
            "List of local ipv6 sids.";
          list ipv6-prefix-sid-local {
            key "prefix algorithm";
            description
              "List of local prefix-sid.";
            uses sr-cmn:ipv6-sid-cfg;
          }
        }
      }
    }
    container global-srgb {
      description
        "Global SRGB configuration.";
      uses sr-cmn:srgb-cfg;
    }
    container srlb {
      description
        "SR Local Block configuration.";
      uses sr-cmn:srlb-cfg;
    }

    list label-blocks {
      config false;
      description
        "List of labels blocks currently
         in use.";
      leaf lower-bound {
        type uint32;
        description
          "Lower bound of the label block.";
      }
      leaf upper-bound {
        type uint32;
        description
          "Upper bound of the label block.";
      }
      leaf size {
        type uint32;
        description
          "Number of indexes in the block.";
```

```
        }
        leaf free {
          type uint32;
          description
            "Number of indexes free in the block.";
        }
        leaf used {
          type uint32;
          description
            "Number of indexes used in the block.";
        }
        leaf scope {
          type enumeration {
            enum "global" {
              description
                "Global sid.";
            }
            enum "local" {
              description
                "Local sid.";
            }
          }
          description
            "Scope of this label block.";
        }
      }
      container sid-list {
        config false;
        description
          "List of prefix and SID associations.";
        list sid {
          key "target sid source source-protocol binding-type";
          ordered-by system;
          description
            "Binding.";
          leaf target {
            type string;
            description
              "Defines the target of the binding.
               It can be a prefix or something else.";
          }
          leaf sid {
            type uint32;
            description
              "Index associated with the prefix.";
          }
          leaf algorithm {
            type uint8;
```

```
            description
              "Algorithm to be used for the prefix
               SID.";
          }
          leaf source {
            type inet:ip-address;
            description
              "IP address of the router than own
               the binding.";
          }
          leaf used {
            type boolean;
            description
              "Defines if the binding is used
               in forwarding plane.";
          }
          leaf source-protocol {
            type leafref {
              path "/rt:routing/rt:control-plane-protocols/"
                 + "rt:control-plane-protocol/rt:name";
            }
            description
              "Rtg protocol that owns the binding";
          }
          leaf binding-type {
            type enumeration {
              enum "prefix-sid" {
                description
                  "Binding is learned from
                   a prefix SID.";
              }
              enum "binding-tlv" {
                description
                  "Binding is learned from
                   a binding TLV.";
              }
            }
            description
              "Type of binding.";
          }
          leaf scope {
            type enumeration {
              enum "global" {
                description
                  "Global sid.";
              }
              enum "local" {
                description
```

```
                  "Local sid.";
              }
            }
            description
              "The sid is local or global.";
          }
        }
      }
    }
  }

  notification segment-routing-global-srgb-collision {
    description
      "This notification is sent when received SRGB blocks from
       a router conflict.";
    list srgb-collisions {
      description
        "List of SRGB blocks that conflict.";
      leaf lower-bound {
        type uint32;
        description
          "Lower value in the block.";
      }
      leaf upper-bound {
        type uint32;
        description
          "Upper value in the block.";
      }
      leaf routing-protocol {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
             + "rt:control-plane-protocol/rt:name";
        }
        description
          "Routing protocol reference that received the event.";
      }
      leaf originating-rtr-id {
        type router-id;
        description
          "Originating router id of this SRGB block.";
      }
    }
  }
  notification segment-routing-global-sid-collision {
    description
      "This notification is sent when a new mapping is learned
       , containing mapping
       where the SID is already used.
```

```
       The notification generation must be throttled with at least
       a 5 second gap. ";
    leaf received-target {
      type string;
      description
        "Target received in the controlplane that
                          caused SID collision.";
    }
    leaf new-sid-rtr-id {
      type router-id;
      description
        "Router Id that advertising the conflicting SID.";
    }
    leaf original-target {
      type string;
      description
        "Target already available in database that have the same SID
         as the received target.";
    }
    leaf original-sid-rtr-id {
      type router-id;
      description
        "Original router ID that advertised the conflicting SID.";
    }
    leaf index {
      type uint32;
      description
        "Value of the index used by two different prefixes.";
    }
    leaf routing-protocol {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols/"
           + "rt:control-plane-protocol/rt:name";
      }
      description
        "Routing protocol reference that received the event.";
    }
  }
  notification segment-routing-index-out-of-range {
    description
      "This notification is sent when a binding
       is received, containing a segment index
       which is out of the local configured ranges.
       The notification generation must be throttled with at least
       a 5 second gap. ";
    leaf received-target {
      type string;
      description
```

```
        "Target received in the controlplane
                        that caused SID collision.";
    }
    leaf received-index {
      type uint32;
      description
        "Value of the index received.";
    }
    leaf routing-protocol {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols/"
           + "rt:control-plane-protocol/rt:name";
      }
      description
        "Routing protocol reference that received the event.";
    }
  }
}
<CODE ENDS>
```

8. Security Considerations

TBD.

9. Acknowledgements

Authors would like to thank Derek Yeung, Acee Lindem, Greg Hankins, Hannes Gredler, Uma Chunduri, Jeffrey Zhang, Shradda Hedge, Les Ginsberg for their contributions.

10. IANA Considerations

TBD.

11. References

11.1. Normative References

[I-D.ietf-isis-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling MSD (Maximum SID Depth) using IS-IS", draft-ietf-isis-segment-routing-msd-04 (work in progress), June 2017.

[I-D.ietf-ospf-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling MSD (Maximum SID Depth) using OSPF", draft-ietf-ospf-segment-routing-msd-05 (work in progress), June 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-12 (work in progress), June 2017.

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

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.

11.2. Informative References

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-06 (work in progress), February 2018.

Authors' Addresses

Stephane Litkowski
Orange Business Service

Email: stephane.litkowski@orange.com

Yingzhen Qu
Huawei

Email: yingzhen.qu@huawei.com

Pushpasis Sarkar
Individual

Email: pushpasis.ietf@gmail.com

Jeff Tantsura
Nuage Networks

Email: jefftant.ietf@gmail.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 19, 2019

S. Litkowski
Orange Business Service
Y. Qu
Huawei
P. Sarkar
Individual
J. Tantsura
Apstra
A. Lindem
Cisco

February 15, 2019

YANG Data Model for Segment Routing
draft-ietf-spring-sr-yang-11

Abstract

This document defines a YANG data model ([RFC6020], [RFC7950]) for segment routing ([RFC8402]) configuration and operation. This YANG model is intended to be used on network elements to configure or operate segment routing. This document defines also generic containers that SHOULD be reused by IGP protocol modules to support segment routing.

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 August 19, 2019.

Table of Contents

1. Introduction

This document defines a YANG data model for segment routing configuration and operation. This document does not define the IGP extensions to support segment routing but defines generic groupings that SHOULD be reused by IGP extension modules. The reason of this design choice is to not require implementations to support all IGP extensions. For example, an implementation may support IS-IS extension but not OSPF.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Terminology and Notation

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].

2.1. Tree diagram

Tree diagrams used in this document follow the notation defined in [RFC8340].

2.2. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
if	ietf-interfaces	[RFC8343]
rt	ietf-routing	[RFC8349]
rt-types	ietf-routing-types	[RFC8294]
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and Corresponding YANG Modules

3. Design of the Data Model

As the module definition is just starting, it is expected that there will be changes as the module matures.

```
module: ietf-segment-routing
  augment /rt:routing:
    +--rw segment-routing
       +--rw transport-type?      identityref
       +--ro node-capabilities
       |  +--ro transport-planes* [transport-plane]
       |  |  +--ro transport-plane    identityref
       |  +--ro entropy-readable-label-depth?   uint8
       +--rw msd {max-sid-depth}?
       |  +--rw node-msd?   uint8
       |  +--rw link-msd
       |     +--rw link-msds* [interface]
```

```
|        +--rw interface    if:interface-ref
|        +--rw msd?         uint8
+--rw bindings
|  +--rw mapping-server {mapping-server}?
|  |  +--rw policy* [name]
|  |     +--rw name       string
|  |     +--rw entries
|  |        +--rw mapping-entry* [prefix algorithm]
|  |           +--rw prefix        inet:ip-prefix
|  |           +--rw value-type?   enumeration
|  |           +--rw start-sid     uint32
|  |           +--rw range?        uint32
|  |           +--rw algorithm     identityref
|  +--rw connected-prefix-sid-map
|  |  +--rw connected-prefix-sid* [prefix algorithm]
|  |     +--rw prefix               inet:ip-prefix
|  |     +--rw value-type?          enumeration
|  |     +--rw start-sid            uint32
|  |     +--rw range?               uint32
|  |     +--rw algorithm            identityref
|  |     +--rw last-hop-behavior?   enumeration
|  |                                {sid-last-hop-behavior}?
|  +--rw local-prefix-sid
|     +--rw local-prefix-sid* [prefix algorithm]
|        +--rw prefix        inet:ip-prefix
|        +--rw value-type?   enumeration
|        +--rw start-sid     uint32
|        +--rw range?        uint32
|        +--rw algorithm     identityref
+--rw global-srgb
|  +--rw srgb* [lower-bound upper-bound]
|     +--rw lower-bound    uint32
|     +--rw upper-bound    uint32
+--rw srlb
|  +--rw srlb* [lower-bound upper-bound]
|     +--rw lower-bound    uint32
|     +--rw upper-bound    uint32
+--ro label-blocks*
|  +--ro lower-bound?   uint32
|  +--ro upper-bound?   uint32
|  +--ro size?          uint32
|  +--ro free?          uint32
|  +--ro used?          uint32
|  +--ro scope?         enumeration
+--ro sid-list
   +--ro sid* [target sid source source-protocol binding-type]
      +--ro target             string
      +--ro sid                uint32
```

```
               +--ro algorithm?         uint8
               +--ro source             inet:ip-address
               +--ro used?              boolean
               +--ro source-protocol    -> /rt:routing
                                           /control-plane-protocols
                                           /control-plane-protocol/name
               +--ro binding-type       enumeration
               +--ro scope?             enumeration

     notifications:
       +---n segment-routing-global-srgb-collision
       |  +--ro srgb-collisions*
       |     +--ro lower-bound?          uint32
       |     +--ro upper-bound?          uint32
       |     +--ro routing-protocol?     -> /rt:routing
       |                                    /control-plane-protocols
       |                                    /control-plane-protocol/name
       |     +--ro originating-rtr-id?   router-id
       +---n segment-routing-global-sid-collision
       |  +--ro received-target?       string
       |  +--ro new-sid-rtr-id?        router-id
       |  +--ro original-target?       string
       |  +--ro original-sid-rtr-id?   router-id
       |  +--ro index?                 uint32
       |  +--ro routing-protocol?      -> /rt:routing
       |                                  /control-plane-protocols
       |                                  /control-plane-protocol/name
       +---n segment-routing-index-out-of-range
          +--ro received-target?    string
          +--ro received-index?     uint32
          +--ro routing-protocol?   -> /rt:routing
                                       /control-plane-protocols
                                       /control-plane-protocol/name
```

4. Configuration

This module augments the "/rt:routing:" with a segment-routing container. This container defines all the configuration parameters related to segment-routing.

The segment-routing configuration is split in global configuration and interface configuration.

The global configuration includes :

- segment-routing transport type : The underlying transport type for segment routing. The version of the model limits the transport

type to an MPLS dataplane. The transport-type is only defined once for a particular routing-instance and is agnostic to the control plane used. Only a single transport-type is supported in this version of the model.

- bindings : Defines prefix to SID mappings. The operator can control advertisement of Prefix-SID independently for IPv4 and IPv6. Two types of mappings are available :
  - Mapping-server : maps non local prefixes to a segment ID. Configuration of bindings does not automatically allow advertisement of those bindings. Advertisement must be controlled by each routing-protocol instance (see Section 5). Multiple mapping policies may be defined.
  - Connected prefixes : maps connected prefixes to a segment ID. Advertisement of the mapping will be done by IGP when enabled for segment routing (see Section 5). The SID value can be expressed as an index (default), or an absolute value. The "last-hop-behavior" configuration dictates the PHP behavior: "explicit-null", "php", or "non-php".
- SRGB (Segment Routing Global Block): Defines a list of label blocks represented by a pair of lower-bound/upper-bound labels. The SRGB is also agnostic to the control plane used. So all routing-protocol instance will have to advertise the same SRGB.
- SRLB (Segment Routing Local Block): Defines a list of label blocks represented by a pair of lower-bound/upper-bound labels, reserved for lcoal SIDs.

5. IGP Control plane configuration

Support of segment-routing extensions for a particular IGP control plane is done by augmenting routing-protocol configuration with segment-routing extensions. This augmentation SHOULD be part of separate YANG modules in order to not create any dependency for implementations to support all protocol extensions.

This module defines groupings that SHOULD be used by IGP segment routing modules.

The "controlplane-cfg" grouping defines the generic global configuration for the IGP.

The "enabled" leaf enables segment-routing extensions for the routing-protocol instance.

The "bindings" container controls the routing-protocol instance's advertisement of local bindings and the processing of received bindings.

5.1. IGP interface configuration

The interface configuration is part of the "igp-interface-cfg" grouping and includes Adjacency SID properties.

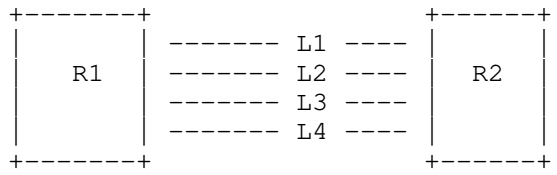
5.1.1. Adjacency SID properties

5.1.1.1. Bundling

This section is a first proposal on how to use S-bit in Adj-SID to create bundles. Authors would like to trigger discussion based on this first proposal.

In case of parallel IP links between routers, an additional Adjacency SID may be advertised representing more than one adjacency (i.e., a bundle of adjacencies). The "advertise-adj-group-sid" configuration controls whether or not an additional adjacency SID is advertised.

The "advertise-adj-group-sid" would be a list of "group-id". The "group-id" will permit to identify interfaces that must be bundled together.



In the figure above, R1 and R2 are interconnected by four links. A routing protocol adjacency is established on each link. Operator would like to create segment-routing Adj-SID that represent some bundles of links. We can imagine two different bundles : L1/L2 and L2/L3. To achieve this behavior, the service provider will configure a "group-id" X for both interfaces L1 and L2 and a "group-id" Y for both interfaces L3 and L3. This will result in R1 advertising an additional Adj-SID for each adjacency, for example a Adj-SID with S flag set and value of 400 will be added to L1 and L2. A Adj-SID with S flag set and value of 500 will be added to L3 and L4. As L1/L2 and L3/L4 does not share the same "group-id", a different SID value will be allocated.

5.1.1.2. Protection

The "advertise-protection" defines how protection for an interface is advertised. It does not control the activation or deactivation of protection. If the "single" option is used, a single Adj-SID will be advertised for the interface. If the interface is protected, the B-Flag for the Adj-SID advertisement will be set. If the "dual" option is used and if the interface is protected, two Adj-SIDs will be advertised for the interface adjacencies. One Adj-SID will always have the B-Flag set and the other will have the B-Flag clear. This option is intended to be used in the case of traffic engineering where a path must use either protected segments or non-protected segments.

6. States

The operational states contains information reflecting the usage of allocated SRGB labels.

It also includes a list of all global SIDs, their associated bindings, and other information such as the source protocol and algorithm.

7. Notifications

The model defines the following notifications for segment-routing.

- segment-routing-global-srgb-collision: Rasied when a control plan advertised SRGB blocks have conflicts.

- segment-routing-global-sid-collision: Raised when a control plane advertised index is already associated with another target (in this version, the only defined targets are IPv4 and IPv6 prefixes).

- segment-routing-index-out-of-range: Raised when a control plane advertised index fall outside the range of SRGBs configured for the network device.

8. YANG Module

The following RFCs and drafts are not referenced in the document text but are referenced in the ietf-segment-rouuting-common.yang and/or ietf-segment-routing.yang module: [RFC6991], [RFC8294], and [RFC8476].

```
<CODE BEGINS> file "ietf-segment-routing-common@2019-02-15.yang"
module ietf-segment-routing-common {
```

```
    namespace "urn:ietf:params:xml:ns:yang:ietf-segment-routing-common";
    prefix sr-cmn;

    import ietf-inet-types {
      prefix inet;
    }

    organization
      "IETF SPRING - SPRING Working Group";

    contact
      "WG Web:   <http://tools.ietf.org/wg/spring/>
       WG List:  <mailto:spring@ietf.org>

       Editor:    Stephane Litkowski
                  <mailto:stephane.litkowski@orange.com>
       Editor:    Yingzhen Qu
                  <mailto:yingzhen.qu@huawei.com>

       Author:    Acee Lindem
                  <mailto:acee@cisco.com>
       Author:    Pushpasis Sarkar
                  <mailto:pushpasis.ietf@gmail.com>
       Author:    Jeff Tantsura
                  <jefftant.ietf@gmail.com>

      ";
    description
      "The YANG module defines a collection of types and groupings for
       Segment routing.

       Copyright (c) 2017 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX;
       see the RFC itself for full legal notices.";

    reference "RFC XXXX";

    revision 2019-02-15 {
      description
```

```
        "
        * Addressed YANG Doctor's review comments;
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }

    revision 2018-06-25 {
      description
        "
        * Renamed readable-label-stack-depth to
        * entropy-readable-label-depth;
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }
    revision 2017-07-01 {
      description
        "
         *Conform to RFC6087BIS Appendix C
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }
    revision 2017-03-10 {
      description
        "
         * Add support of SRLB
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }
    revision 2016-10-28 {
      description
        "
         * Add support of MSD (Maximum SID Depth)
         * Update contact info
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }
    revision 2016-10-24 {
      description
        "Initial";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }

    feature sid-last-hop-behavior {
      description
        "Configurable last hop behavior.";
    }

    identity segment-routing-transport {
```

```
      description
        "Base identity for segment routing transport.";
    }

    identity segment-routing-transport-mpls {
      base segment-routing-transport;
      description
        "This identity represents MPLS transport for segment
         routing.";
    }

    identity segment-routing-transport-ipv6 {
      base segment-routing-transport;
      description
        "This identity represents IPv6 transport for segment
         routing.";
    }

    identity prefix-sid-algorithm {
      description
        "Base identity for prefix-sid algorithm.";
    }

    identity prefix-sid-algorithm-shortest-path {
      base prefix-sid-algorithm;
      description
        "The default behavior of prefix-sid algorithm.";
    }

    identity prefix-sid-algorithm-strict-spf {
      base prefix-sid-algorithm;
      description
        "This algorithm mandates that the packet is forwared
         according to ECMP-aware SPF algorithm.";
    }

    grouping srlr {
      description
        "Grouping for SR Label Range configuration.";
      leaf lower-bound {
          type uint32;
          description
            "Lower value in the block.";
        }
      leaf upper-bound {
          type uint32;
          description
            "Upper value in the block.";
```

```
      }
    }

    grouping srgb {
      description
        "Grouping for SR Global Label range.";
      list srgb {
        key "lower-bound upper-bound";
        ordered-by user;
        description
          "List of global blocks to be
           advertised.";
        uses srlr;
      }
    }

    grouping srlb {
      description
        "Grouping for SR Local Block range.";
      list srlb {
        key "lower-bound upper-bound";
        ordered-by user;
        description
          "List of SRLBs.";
        uses srlr;
      }
    }

    grouping sid-value-type {
      description
        "Defines how the SID value is expressed.";
      leaf value-type {
        type enumeration {
          enum "index" {
            description
              "The value will be
               interpreted as an index.";
          }
          enum "absolute" {
            description
              "The value will become
               interpreted as an absolute
               value.";
          }
        }
        default "index";
        description
          "This leaf defines how value
```

```
            must be interpreted.";
        }
      }

      grouping prefix-sid {
        description
          "This grouping defines cfg of prefix SID.";
        leaf prefix {
          type inet:ip-prefix;
          description
            "connected prefix sid.";
        }
        uses prefix-sid-attributes;
      }

      grouping ipv4-sid {
        description
          "This grouping defines ipv4 prefix SID.";
        leaf prefix {
          type inet:ipv4-prefix;
          description
            "connected prefix sid.";
        }
        uses prefix-sid-attributes;
      }
      grouping ipv6-sid {
        description
          "This grouping defines ipv6 prefix SID.";
        leaf prefix {
          type inet:ipv6-prefix;
          description
            "connected prefix sid.";
        }
        uses prefix-sid-attributes;
      }

      grouping last-hop-behavior {
        description
          "Defines last hop behavior";
        leaf last-hop-behavior {
          if-feature "sid-last-hop-behavior";
          type enumeration {
            enum "explicit-null" {
              description
                "Use explicit-null for the SID.";
            }
            enum "no-php" {
              description
```

```
            "Do no use PHP for the SID.";
        }
        enum "php" {
          description
            "Use PHP for the SID.";
        }
      }
      description
        "Configure last hop behavior.";
    }
  }

  grouping node-capabilities {
    description
      "Containing SR node capabilities.";
    container node-capabilities {
      config false;
      description
        "Shows the SR capability of the node.";
      list transport-planes {
        key "transport-plane";
        description
          "List of supported transport planes.";
        leaf transport-plane {
          type identityref {
            base segment-routing-transport;
          }
          description
            "Transport plane supported";
        }
      }
      leaf entropy-readable-label-depth {
        type uint8;
        description
          "Maximum label statck depth that
           the router can read. ";
      }
    }
  }

  grouping prefix-sid-attributes {
    description
      "Containing SR attributes for a prefix.";
    uses sid-value-type;
    leaf start-sid {
      type uint32;
      mandatory true;
      description
```

```
          "Value associated with
           prefix. The value must
           be interpreted in the
           context of value-type.";
      }
      leaf range {
        type uint32;
        description
          "Describes how many SIDs could be
           allocated.";
      }
      leaf algorithm {
        type identityref {
          base prefix-sid-algorithm;
        }
        description
          "Prefix-sid algorithm.";
      }
    }
  }
  <CODE ENDS>
  <CODE BEGINS> file "ietf-segment-routing@2019-02-15.yang"
  module ietf-segment-routing {
    namespace "urn:ietf:params:xml:ns:yang:ietf-segment-routing";
    prefix sr;

    import ietf-inet-types {
      prefix inet;
    }
    import ietf-yang-types {
      prefix yang;
    }
    import ietf-routing {
      prefix rt;
    }
    import ietf-interfaces {
      prefix if;
    }
    import ietf-routing-types {
      prefix rt-types;
    }
    import ietf-segment-routing-common {
      prefix sr-cmn;
    }

    organization
      "IETF SPRING - SPRING Working Group";
    contact
```

```
      "WG Web:   <http://tools.ietf.org/wg/spring/>
       WG List:  <mailto:spring@ietf.org>

       Editor:    Stephane Litkowski
                 <mailto:stephane.litkowski@orange.com>
       Editor:    Yingzhen Qu
                 <mailto:yingzhen.qu@huawei.com>

       Author:    Acee Lindem
                 <mailto:acee@cisco.com>
       Author:    Pushpasis Sarkar
                 <mailto:pushpasis.ietf@gmail.com>
       Author:    Jeff Tantsura
                 <jefftant.ietf@gmail.com>

      ";
    description
      "The YANG module defines a generic configuration model for
       Segment routing common across all of the vendor
       implementations.

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

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX;
       see the RFC itself for full legal notices.";

    reference "RFC XXXX";

    revision 2019-02-15 {
      description
        "
        * Addressed YANG Doctor's review comments;
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }

    revision 2018-06-25 {
      description
        "";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
```

```
    }
    revision 2017-07-01 {
      description
        "
         * Implement NMDA model
         *Conform to RFC6087BIS Appendix C
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }

    revision 2017-03-10 {
      description
        "
         * Change global-sid-list to sid-list and add a leaf scope
         * Added support of SRLB
         * Added support of local sids
         * fixed indentations
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }
    revision 2016-10-28 {
      description
        "
         * Add support of MSD (Maximum SID Depth)
         * Update contact info
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }
    revision 2016-10-24 {
      description
        "
         * Moved common SR types and groupings to a seperate module
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }
    revision 2016-07-07 {
      description
        "
         * Add support of prefix-sid algorithm configuration
         * change routing-protocols to control-plane-protocols
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }
    revision 2016-03-17 {
      description
        "
         * Add notification segment-routing-global-srgb-collision
         * Add router-id to segment-routing-global-sid-collision
```

```
         * Remove routing-instance
         * Add typedef router-id
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }
    revision 2015-10-17 {
      description
        "
         * Add per-protocol SRGB config feature
         * Move SRBG config to a grouping
        ";
      reference "RFC XXXX: YANG Data Model for Segment Routing.";
    }
    revision 2015-06-22 {
      description
        "
         * Prefix SID config moved to
         connected-prefix-sid-map in global SR cfg
         rather than IGP.
        ";
      reference "draft-litkowski-spring-sr-yang-01";
    }
    revision 2015-04-23 {
      description
        "
         * Node flag deprecated from prefixSID
         * SR interface cfg moved to protocol
         * Adding multiple binding policies for SRMS
        ";
      reference "";
    }
    revision 2015-02-27 {
      description
        "Initial";
      reference "draft-litkowski-spring-sr-yang-00";
    }

    feature mapping-server {
      description
        "Support of Segment Routing Mapping Server (SRMS).";
    }

    feature protocol-srgb {
      description
        "Support per-protocol Segment Routing Global Block (SRGB)
         configuration.";
    }
```

```
    feature max-sid-depth {
      description
        "Support of signaling MSD (Maximum SID Depth) in IGP.";
    }

    typedef system-id {
      type string {
        pattern "[0-9A-Fa-f]{4}\\.[0-9A-Fa-f]{4}\\.[0-9A-Fa-f]{4}\\.00";
      }
      description
        "This type defines ISIS system id using pattern,
         system id looks like : 0143.0438.AeF0.00";
    }

    typedef router-id {
      type union {
        type system-id;
        type rt-types:router-id;
      }
      description
        "OSPF/BGP router id or ISIS system ID.";
    }

    grouping sr-controlplane {
      description
        "Defines protocol configuration.";
      container segment-routing {
        description
          "segment routing global config.";
        leaf enabled {
          type boolean;
          default "false";
          description
            "Enables segment-routing
             protocol extensions.";
        }
        container bindings {
          description
            "Control of binding advertisement
             and reception.";
          container advertise {
            description
              "Authorize the advertise
               of local mappings in binding TLV.";
            leaf-list policies {
              type string;
              description
                "List of policies to be advertised.";
```

```
            }
          }
          leaf receive {
            type boolean;
            default "true";
            description
              "Authorize the reception and usage
               of binding TLV.";
          }
        }
      }
    }

    grouping igp-interface {
      description
        "Grouping for IGP interface cfg.";
      container segment-routing {
        description
          "container for SR interface cfg.";
        container adjacency-sid {
          description
            "Defines the adjacency SID properties.";
          list advertise-adj-group-sid {
            key "group-id";
            description
              "Control advertisement of S flag.
               Enable to advertise a common Adj-SID
               for parallel links.";
            leaf group-id {
              type uint32;
              description
                "The value is an internal value to identify
                 a group-ID. Interfaces with the same
                 group-ID will be bundled together.";
            }
          }
          leaf advertise-protection {
            type enumeration {
              enum "single" {
                description
                  "A single Adj-SID is associated
                   with the adjacency and reflects
                   the protection configuration.";
              }
              enum "dual" {
                description
                  "Two Adj-SIDs will be associated
                   with the adjacency if interface
```

```
                    is protected. In this case
                    one will be enforced with
                    backup flag set, the other
                    will be enforced to backup flag unset.
                    In case, protection is not configured,
                    a single Adj-SID will be advertised
                    with backup flag unset.";
                }
              }
              description
                "If set, the Adj-SID refers to an
                 adjacency being protected.";
            }
          }
        }
      }

      grouping max-sid-depth {
        description
          "MSD configuration grouping.";
        leaf node-msd {
          type uint8;
          description
            "Node MSD is the lowest MSD supported by the node.";
        }
        container link-msd {
          description
            "Link MSD is a number representing the particular link
             MSD value.";
          list link-msds {
            key "interface";
            description
              "List of link MSDs.";
            leaf interface {
              type if:interface-ref;
              description
                "Name of the interface.";
            }
            leaf msd {
              type uint8;
              description
                "SID depth of the interface associated with the link.";
            }
          }
        }
      }

      augment "/rt:routing" {
```

```
      description
        "This augments routing data model (RFC 8349)
         with segment-routing.";
      container segment-routing {
        description
          "segment routing global config.";
        leaf transport-type {
          type identityref {
            base sr-cmn:segment-routing-transport;
          }
          default "sr-cmn:segment-routing-transport-mpls";
          description
            "Dataplane to be used.";
        }
        uses sr-cmn:node-capabilities;
        container msd {
          if-feature "max-sid-depth";
          description
            "MSD configuration.";
          uses max-sid-depth;
        }
        container bindings {
          description
            "List of bindings.";
          container mapping-server {
            if-feature "mapping-server";
            description
              "Configuration of mapping-server
               local entries.";
            list policy {
              key "name";
              description
                "Definition of mapping policy.";
              leaf name {
                type string;
                description
                  "Name of the mapping policy.";
              }
              container entries {
                description
                  "IPv4/IPv6 mapping entries.";
                list mapping-entry {
                  key "prefix algorithm";
                  description
                    "Mapping entries.";
                  uses sr-cmn:prefix-sid;
                }
              }
```

```
          }
        }
        container connected-prefix-sid-map {
          description
            "Prefix SID configuration.";
          list connected-prefix-sid {
            key "prefix algorithm";
            description
              "List of prefix SID mapped to
               ipv4/ipv6 local prefixes.";
            uses sr-cmn:prefix-sid;
            uses sr-cmn:last-hop-behavior;
          }
        }
        container local-prefix-sid {
          description
            "Local sid configuration.";
          list local-prefix-sid {
            key "prefix algorithm";
            description
              "List of local ipv4/ipv6 prefix-sid.";
            uses sr-cmn:prefix-sid;
          }
        }
      }
      container global-srgb {
        description
          "Global SRGB configuration.";
        uses sr-cmn:srgb;
      }
      container srlb {
        description
          "SR Local Block configuration.";
        uses sr-cmn:srlb;
      }

      list label-blocks {
        config false;
        description
          "List of labels blocks currently
           in use.";
        leaf lower-bound {
          type uint32;
          description
            "Lower bound of the label block.";
        }
        leaf upper-bound {
          type uint32;
```

```
            description
              "Upper bound of the label block.";
          }
          leaf size {
            type uint32;
            description
              "Number of indexes in the block.";
          }
          leaf free {
            type uint32;
            description
              "Number of indexes free in the block.";
          }
          leaf used {
            type uint32;
            description
              "Number of indexes used in the block.";
          }
          leaf scope {
            type enumeration {
              enum "global" {
                description
                  "Global sid.";
              }
              enum "local" {
                description
                  "Local sid.";
              }
            }
            description
              "Scope of this label block.";
          }
        }
        container sid-list {
          config false;
          description
            "List of prefix and SID associations.";
          list sid {
            key "target sid source source-protocol binding-type";
            ordered-by system;
            description
              "Binding.";
            leaf target {
              type string;
              description
                "Defines the target of the binding.
                 It can be a prefix or something else.";
            }
```

```
            leaf sid {
              type uint32;
              description
                "Index associated with the prefix.";
            }
            leaf algorithm {
              type uint8;
              description
                "Algorithm to be used for the prefix
                 SID.";
            }
            leaf source {
              type inet:ip-address;
              description
                "IP address of the router than own
                 the binding.";
            }
            leaf used {
              type boolean;
              description
                "Defines if the binding is used
                 in forwarding plane.";
            }
            leaf source-protocol {
              type leafref {
                path "/rt:routing/rt:control-plane-protocols/"
                   + "rt:control-plane-protocol/rt:name";
              }
              description
                "Rtg protocol that owns the binding";
            }
            leaf binding-type {
              type enumeration {
                enum "prefix-sid" {
                  description
                    "Binding is learned from
                     a prefix SID.";
                }
                enum "binding-tlv" {
                  description
                    "Binding is learned from
                     a binding TLV.";
                }
              }
              description
                "Type of binding.";
            }
            leaf scope {
```

```
              type enumeration {
                enum "global" {
                  description
                    "Global sid.";
                }
                enum "local" {
                  description
                    "Local sid.";
                }
              }
              description
                "The sid is local or global.";
            }
          }
        }
      }
    }

    notification segment-routing-global-srgb-collision {
      description
        "This notification is sent when received SRGB blocks from
         a router conflict.";
      list srgb-collisions {
        description
          "List of SRGB blocks that conflict.";
        leaf lower-bound {
          type uint32;
          description
            "Lower value in the block.";
        }
        leaf upper-bound {
          type uint32;
          description
            "Upper value in the block.";
        }
        leaf routing-protocol {
          type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
               + "rt:control-plane-protocol/rt:name";
          }
          description
            "Routing protocol reference that received the event.";
        }
        leaf originating-rtr-id {
          type router-id;
          description
            "Originating router id of this SRGB block.";
        }
```

```
      }
    }
    notification segment-routing-global-sid-collision {
      description
        "This notification is sent when a new mapping is learned
         , containing mapping
         where the SID is already used.
         The notification generation must be throttled with at least
         a 5 second gap. ";
      leaf received-target {
        type string;
        description
          "Target received in the controlplane that
                              caused SID collision.";
      }
      leaf new-sid-rtr-id {
        type router-id;
        description
          "Router Id that advertising the conflicting SID.";
      }
      leaf original-target {
        type string;
        description
          "Target already available in database that have the same SID
           as the received target.";
      }
      leaf original-sid-rtr-id {
        type router-id;
        description
          "Original router ID that advertised the conflicting SID.";
      }
      leaf index {
        type uint32;
        description
          "Value of the index used by two different prefixes.";
      }
      leaf routing-protocol {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
             + "rt:control-plane-protocol/rt:name";
        }
        description
          "Routing protocol reference that received the event.";
      }
    }
    notification segment-routing-index-out-of-range {
      description
        "This notification is sent when a binding
```

```
         is received, containing a segment index
         which is out of the local configured ranges.
         The notification generation must be throttled with at least
         a 5 second gap. ";
      leaf received-target {
        type string;
        description
          "Target received in the controlplane
                           that caused SID collision.";
      }
      leaf received-index {
        type uint32;
        description
          "Value of the index received.";
      }
      leaf routing-protocol {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
             + "rt:control-plane-protocol/rt:name";
        }
        description
          "Routing protocol reference that received the event.";
      }
    }
  }
  <CODE ENDS>
```

9. Security Considerations

TBD.

10. Acknowledgements

Authors would like to thank Derek Yeung, Acee Lindem, Greg Hankins, Hannes Gredler, Uma Chunduri, Jeffrey Zhang, Shradda Hedge, Les Ginsberg for their contributions.

11. IANA Considerations

TBD.

12. References

12.1. Normative References

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[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>.

[RFC8476] Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling Maximum SID Depth (MSD) Using OSPF", RFC 8476, DOI 10.17487/RFC8476, December 2018, <https://www.rfc-editor.org/info/rfc8476>.

[RFC8491] Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling Maximum SID Depth (MSD) Using IS-IS", RFC 8491, DOI 10.17487/RFC8491, November 2018, <https://www.rfc-editor.org/info/rfc8491>.

12.2. Informative References

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Authors' Addresses

Stephane Litkowski
Orange Business Service

Email: stephane.litkowski@orange.com

Yingzhen Qu
Huawei

Email: yingzhen.qu@huawei.com

Pushpasis Sarkar
Individual

Email: pushpasis.ietf@gmail.com

Jeff Tantsura
Apstra

Email: jefftant.ietf@gmail.com

Acee Lindem
Cisco
301 Mindenhall Way
Cary, NC 27513
US

Email: acee@cisco.com

Network Working Group
Internet-Draft
Updates: 8287 (if approved)
Intended status: Standards Track
Expires: December 30, 2018

F. Iqbal, Ed.
N. Kumar
Z. Ali
C. Pignataro
Cisco

June 28, 2018

Supporting Flexible Algorithm Prefix SIDs in LSP Ping/Traceroute draft-iqbal-spring-mpls-ping-algo-00

Abstract

RFC8287 defines the extensions to MPLS LSP Ping and Traceroute for Segment Routing IGP-Prefix and IGP-Adjacency Segment Identifier (SIDs) with an MPLS data plane. [I-D.ietf-lsr-flex-algo] proposes a mechanism to allow IGPs to compute constraint based path over network and use Segment Routing Prefix-SIDs to steer packets along the constraint-based paths. All Prefix-SIDs associated with the Flexible Algorithm are assigned to the same IPv4/IPv6 Prefix. Any Segment Routing network that uses Flexible Algorithm based path computation needs additional details to be carried in the FEC Stack sub-TLV for FEC validaiton.

This document updates [RFC8287] by modifying IPv4 and IPv6 IGP-Prefix Segment ID FEC sub-TLVs to also include algorithm identification.

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 December 30, 2018.

Table of Contents

1. Introduction

[RFC8287] defines the extensions to MPLS LSP Ping and Traceroute for Segment Routing IGP-Prefix SID and IGP-Adjacency SID with an MPLS data plane. [RFC8287] proposes 3 Target FEC Stack Sub-TLVs to carry this information. [I-D.ietf-lsr-flex-algo] introduces the concept of Flexible Algorithm that allows IGPs (ISIS, OSPFv2 and OSPFv3) to compute constraint-based path over an MPLS network. The constraint-based paths enables the IGP of a router to associate one or more Segment Routing Prefix-SID with a particular Flexible Algorithm. Multiple Flexible Algorithms are assigned to the same IPv4/IPv6 Prefix while each utilizing a different MPLS Prefix SID label.

Existing MPLS Ping/Traceroute machinery for SR Prefix SIDs, defined in [RFC8287], carries prefix, prefix length, and IGP protocol. To correctly identify and validate a Flexible Algorithm Prefix-SID, the validating device also requires algorithm identification to be supplied in the FEC Stack sub-TLV. This document extends SR-IGP IPv4 and IPv6 Prefix SID FECs to validate a particular Flexible Algorithm, while maintaining backwards compatibility with existing implementations of [RFC8287].

1.1. Conventions

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

The term "Must Be Zero" (MBZ) is used in object descriptions for reserved fields. These fields MUST be set to zero when sent and ignored on receipt.

Since this document refers to the MPLS Time to Live (TTL) far more frequently than the IP TTL, the authors have chosen the convention of using the unqualified "TTL" to mean "MPLS TTL" and using "IP TTL" for the TTL value in the IP header.

2. Motivation

In presence of Flexible Algorithms, a single IGP Prefix may be associated with zero or more IGP Prefix SIDs in addition to the default (Shortest Path First) Prefix SID. Each Prefix SID will have a distinct Prefix SID label and may possibly have a distinct set of next-hops based on associated constraint-based path calculation criteria. This means that to reach the same destination, Flexible Algorithm based IGP-Prefix SID may take a different path than default IGP Prefix SID algorithm.

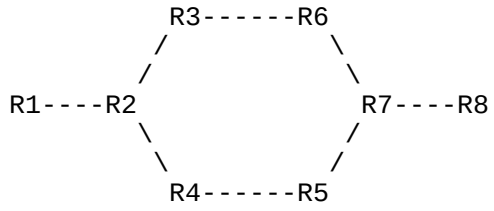


Figure above, which is a simplification of the diagram used in [RFC8287] illustrates this point through an example. Node Segment IDs for R1, R2, R3, R4, R5, R6, R7, and R8 for the default algorithm are 5001, 5002, 5003, 5004, 5005, 5006, 5007, and 5008, respectively. Nodes R1, R2, R4, R5, R7, and R8 also participate in Flexible Algorithm 128. Their corresponding Node Segment IDs for the algorithm are 5801, 5802, 5804, 5805, 5807, and 5808, respectively.

Now consider an MPLS LSP Traceroute request to validate the path to reach node R8 through Flexible Algorithm 128. The TTL of the first echo request packet expires at node R2 with incoming label 5808. Node R2 attempts to validate IGP-Prefix SID Target FEC stack sub-TLV from the echo request. However, this TFS sub-TLV does not contain information identifying the algorithm. As a result, R2 will attempt validation with default algorithm which expects the echo packet to arrive with Prefix SID label 5008. The validation fails, and node R2 responds with error code 10 resulting in a false negative.

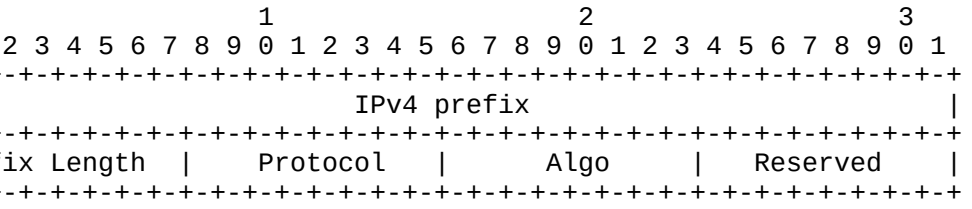
Carrying algorithm identification in the Target FEC Stack sub-TLV of MPLS echo request will help avoid such false negatives. It will also help detect forwarding deviations such as when the packet for a particular destination is incorrectly forwarded to a device that is participating in the default algo but does not participate in a given Flexible Algorithm.

3. Algorithm Identification for IGP-Prefix SID Sub-TLVs

Section 5 of [RFC8287] defines 3 different Segment ID Sub-TLVs that will be included in Target FEC Stack TLV defined in [RFC8029]. This section updates IPv4 IGP-Prefix Segment ID Sub-TLV and IPv6 IGP-Prefix Segment ID Sub-TLV to also include an additional field identifying the algorithm.

3.1. IPv4 IGP-Prefix Segment ID Sub-TLV

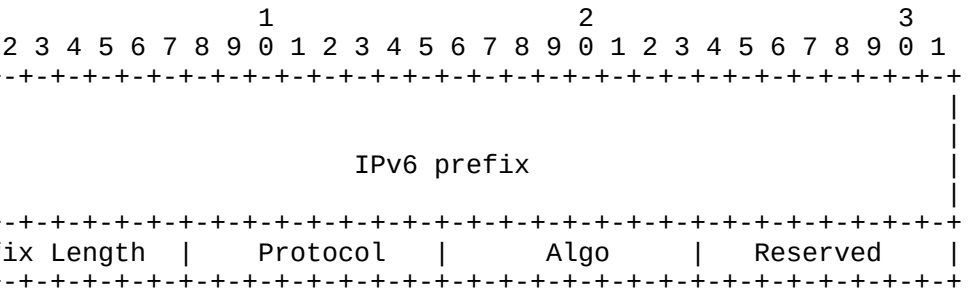
The Sub-TLV format for IPv4 IGP-Prefix Segment ID MUST be set as shown in the below TLV format:



Algo field must be set to 0 if the default algorithm is used. Algo field is set to 1 if Strict Shortest Path First (Strict-SPF) algorithm is used. For Flex-Algo, the Algo field must be set with the algorithm value (values can be 128-255).

3.2. IPv6 IGP-Prefix Segment ID Sub-TLV

The Sub-TLV format for IPv6 IGP-Prefix Segment ID MUST be set as shown in the below TLV format:



Algo field must be set to 0 if the default algorithm is used. Algo field is set to 1 if Strict Shortest Path First (Strict-SPF) algorithm is used. For Flex-Algo, the Algo field must be set with the algorithm value (values can be 128-255).

4. Procedures

4.1. Initiator Node Procedures

A node initiating LSP echo request packet for the Node Segment ID MUST identify and include the algorithm associated with the IGP Prefix SID in the Target FEC Stack sub-TLV. If the initiating node is not aware of the algorithm, the default algorithm (id 0) of Shortest Path First is assumed.

4.2. Responder Node Procedures

This section updates the procedures defined in Section 7.4 of [RFC8287] for IPv4/IPv6 IGP Prefix SID FEC. If the algorithm is 0, the procedures from [RFC8287] do not require any change. For any other algorithm value, if the responding node is validating the FEC stack, it MUST also validate the IGP Prefix SID advertisement for the algorithm defined in Algo field.

If the responding node is including IGP Prefix SID FEC in the FEC stack due to FEC Stack Change operation, it MUST also include algorithm associated with the Prefix SID.

5. IANA Considerations

This document does not introduce any IANA considerations.

6. Security Considerations

This document updates [RFC8287] and does not introduce any security considerations.

7. Acknowledgements

TBA.

8. Contributors

TBA

9. References

9.1. Normative References

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

[RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <https://www.rfc-editor.org/info/rfc8029>.

[RFC8287] Kumar, N., Ed., Pignataro, C., Ed., Swallow, G., Akiya, N., Kini, S., and M. Chen, "Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes", RFC 8287, DOI 10.17487/RFC8287, December 2017, <https://www.rfc-editor.org/info/rfc8287>.

9.2. Informative References

[I-D.ietf-lsr-flex-algo] Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., Gulko, A., "IGP Flexible Algorithm", <http://tools.ietf.org/html/ietf-lsr-flex-algo>.

Authors' Addresses

Faisal Iqbal (editor)
Cisco Systems, Inc.

Email: faiqbal@cisco.com

Nagendra Kumar
Cisco Systems, Inc.

Email: naikumar@cisco.com

Zafar Ali
Cisco Systems, Inc.

Email: zali@cisco.com

Carlos Pignataro
Cisco Systems, Inc.

Email: cpignata@cisco.com

Network Working Group
Internet-Draft
Updates: 8287 (if approved)
Intended status: Standards Track
Expires: April 26, 2019

F. Iqbal, Ed.
N. Kumar
Z. Ali
C. Pignataro
Cisco

October 23, 2018

LSP Ping/Traceroute for Prefix SID in Presence of Multi-Algorithm/Multi-Topology Networks
draft-iqbal-spring-mpls-ping-algo-01

Abstract

[RFC8287] defines the extensions to MPLS LSP Ping and Traceroute for Segment Routing IGP-Prefix and IGP-Adjacency Segment Identifier (SIDs) with an MPLS data plane. The machinery defined in [RFC8287] works well in single topology, single algorithm deployments where each Prefix SID is only associated with a single IP prefix. In multi-topology networks, or networks deploying multiple algorithms for the same IP Prefix, MPLS echo request needs to carry additional information in the Target FEC Stack sub-TLVs to properly validate IGP Prefix SID.

This document updates [RFC8287] by modifying IPv4 and IPv6 IGP-Prefix Segment ID FEC sub-TLVs to also include algorithm identification while maintaining backwards compatibility. This document also introduces new Target FEC Stack sub-TLVs for Prefix SID validation in multi-topology networks.

Table of Contents

1. Introduction

[RFC8287] defines the extensions to MPLS LSP Ping and Traceroute for Segment Routing IGP-Prefix SID and IGP-Adjacency SID with an MPLS data plane. [RFC8287] proposes 3 Target FEC Stack Sub-TLVs to carry this information. [I-D.ietf-lsr-flex-algo] introduces the concept of Flexible Algorithm that allows IGPs (ISIS, OSPFv2 and OSPFv3) to compute constraint-based path over an MPLS network. The constraint-based paths enables the IGP of a router to associate one or more Segment Routing Prefix-SID with a particular Flexible Algorithm, and steer packets along the constraint-based paths. Multiple Flexible Algorithms are assigned to the same IPv4/IPv6 Prefix while each utilizing a different MPLS Prefix SID label. Similarly, operators may deploy same IP prefix across multiple topologies in the network using IGP Multi-topology ID (MT-ID). As Flexible-Algorithm based deployments in particular, and multi-topology networks in general, become more common, existing OAM machinery requires updates to correctly diagnose network faults.

Segment Routing architecture [RFC8402] defines the context for IGP Prefix SID as a unique tuple comprised of prefix, topology, and algorithm>. Existing MPLS Ping/Traceroute machinery for SR Prefix SIDs, defined in [RFC8287], carries prefix, prefix length, and IGP protocol. To correctly identify and validate a Prefix-SID, the validating device also requires algorithm and topology identification to be supplied in the FEC Stack sub-TLV. This document extends SR-IGP IPv4 and IPv6 Prefix SID FECs to validate a particular algorithm in a single-topology network, while maintaining backwards compatibility with existing implementations of [RFC8287]. It also introduces new Target FEC Stack sub-TLVs to perform MPLS Ping and Traceroute for IGP Prefix SIDs in multi-topology, multi-algorithm deployments.

2. Conventions

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

The term "Must Be Zero" (MBZ) is used in object descriptions for reserved fields. These fields MUST be set to zero when sent and ignored on receipt.

Since this document refers to the MPLS Time to Live (TTL) far more frequently than the IP TTL, the authors have chosen the convention of using the unqualified "TTL" to mean "MPLS TTL" and using "IP TTL" for the TTL value in the IP header.

3. Motivation

In presence of multiple algorithms, a single IGP Prefix may be associated with zero or more IGP Prefix SIDs in addition to the default (Shortest Path First) Prefix SID. Each Prefix SID will have a distinct Prefix SID label and may possibly have a distinct set of next-hops based on associated constraint-based path calculation criteria. This means that to reach the same destination, an non-default algorithm IGP-Prefix SID may take a different path than default IGP Prefix SID algorithm.

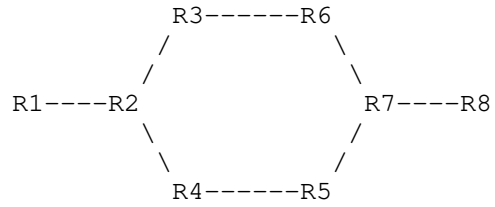


Figure above, which is a simplification of the diagram used in [RFC8287] illustrates this point through an example. Node Segment IDs for R1, R2, R3, R4, R5, R6, R7, and R8 for the default algorithm are 5001, 5002, 5003, 5004, 5005, 5006, 5007, and 5008, respectively. Nodes R1, R2, R4, R5, R7, and R8 also participate in Flexible Algorithm 128. Their corresponding Node Segment IDs for the algorithm are 5801, 5802, 5804, 5805, 5807, and 5808, respectively.

Now consider an MPLS LSP Traceroute request to validate the path to reach node R8 through Flexible Algorithm 128. The TTL of the first echo request packet expires at node R2 with incoming label 5808. Node R2 attempts to validate IGP-Prefix SID Target FEC stack sub-TLV from the echo request. However, this TFS sub-TLV does not contain information identifying the algorithm. As a result, R2 will attempt validation with default algorithm which expects the echo packet to arrive with Prefix SID label 5008. The validation fails, and node R2 responds with error code 10 resulting in a false negative.

Carrying algorithm identification in the Target FEC Stack sub-TLV of MPLS echo request will help avoid such false negatives. It will also help detect forwarding deviations such as when the packet for a particular destination is incorrectly forwarded to a device that is participating in the default algo but does not participate in a given Flexible Algorithm.

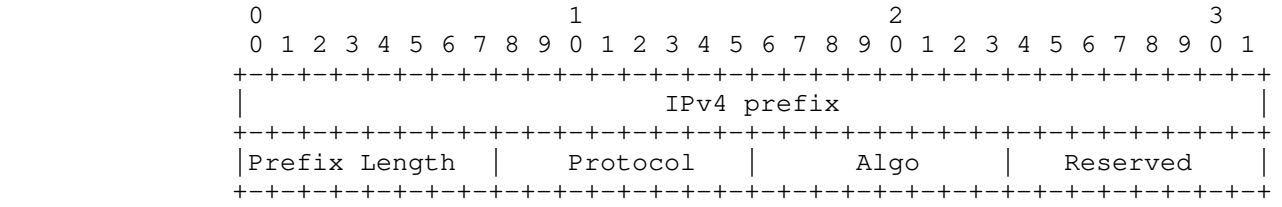
The above problem statement can also be extended to apply in Multi-Topology networks. In such networks, the Target FEC Stack sub-TLV MUST carry Multi-Topology ID (MT-ID) in addition to prefix, its length, IGP identification, and algorithm.

4. Algorithm Identification for IGP-Prefix SID Sub-TLVs

Section 5 of [RFC8287] defines 3 different Segment ID Sub-TLVs that will be included in Target FEC Stack TLV defined in [RFC8029]. This section updates IPv4 IGP-Prefix Segment ID Sub-TLV and IPv6 IGP-Prefix Segment ID Sub-TLV to also include an additional field identifying the algorithm.

4.1. IPv4 IGP-Prefix Segment ID Sub-TLV

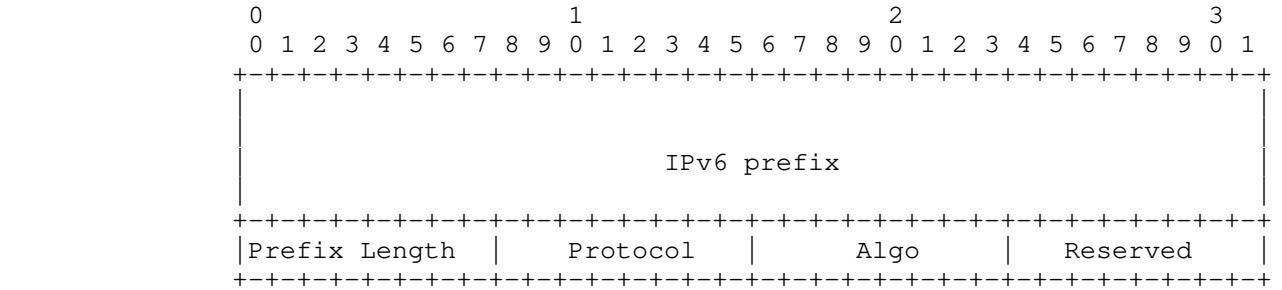
The Sub-TLV format for IPv4 IGP-Prefix Segment ID MUST be set as shown in the below TLV format:



Algo field MUST be set to 0 if the default algorithm is used. Algo field is set to 1 if Strict Shortest Path First (Strict-SPF) algorithm is used. For Flex-Algo, the Algo field MUST be set with the algorithm value (values can be 128-255).

4.2. IPv6 IGP-Prefix Segment ID Sub-TLV

The Sub-TLV format for IPv6 IGP-Prefix Segment ID MUST be set as shown in the below TLV format:



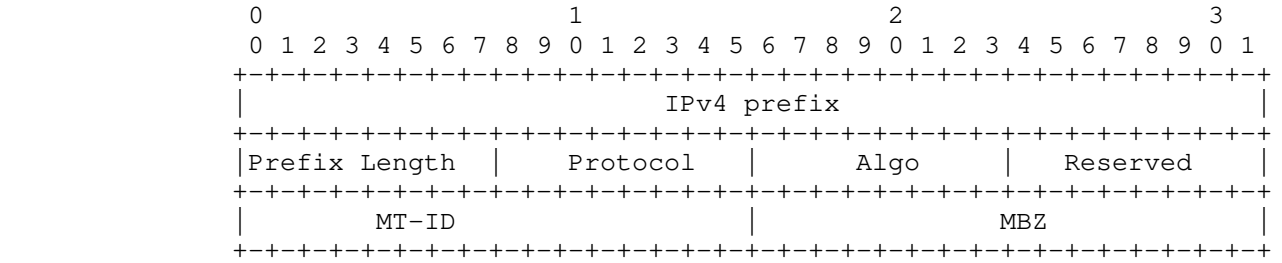
Algo field MUST be set to 0 if the default algorithm is used. Algo field is set to 1 if Strict Shortest Path First (Strict-SPF) algorithm is used. For Flex-Algo, the Algo field MUST be set with the algorithm value (values can be 128-255).

5. Multi-topology Support for IGP Prefix SID

IGP Prefix SID TLVs defined above assume a single-topology network for path validation. For Multi-Topology networks, this section introduces new Multi-Topology IGP IPv4 Prefix SID and Multi-Topology IGP IPv6 Prefix SID sub-TLVs in the Target FEC Stack TLV of MPLS echo request. These sub-TLVs carry MT-ID for OSPF and IS-IS protocols as specified in [RFC4915] and [RFC5120] respectively.

5.1. Multi-topology IPv4 IGP-Prefix Segment ID Sub-TLV

The Sub-TLV format for Multi-topology IPv4 IGP-Prefix Segment ID MUST be set as shown in the below TLV format:

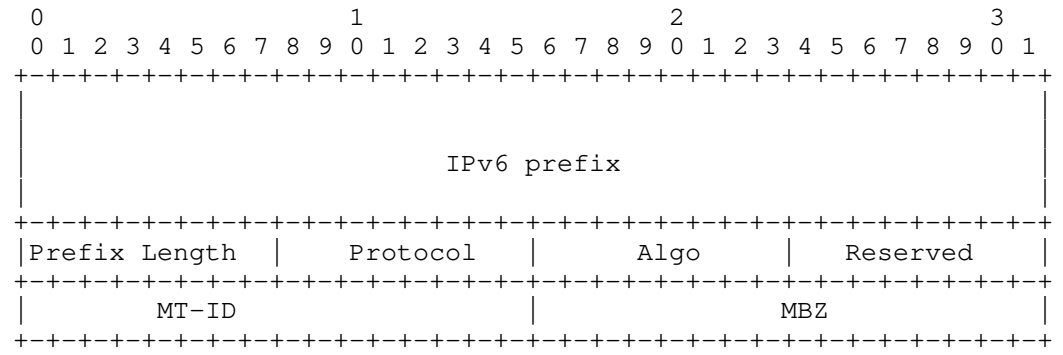


MT-ID identifies the Multi-Topology ID associated with the Prefix SID. MT-ID is set in trailing 12 bits of the field when the Protocol is set to IS-IS. Leading 4-bits of the MT-ID MUST be all zeroes for IS-IS. MT-ID is set to trailing 8 bits when the protocol is specified as OSPF. The leading octet MUST be set to all zeroes for OSPF. MBZ MUST be set to all zeroes.

The Protocol field MUST be set 1 if the responder MUST perform FEC validation using OSPF as the IGP protocol and MT-ID is an OSPF Multi-Topology ID. Protocol is set to 2 if the responder MUST perform FEC validation using IS-IS as the IGP protocol, and the MT-ID is IS-IS Multi-Topology ID. Protocol MUST not be set to 0 when using Multi-Topology IPv4 IGP Prefix SID sub-TLV.

5.2. Multi-Topology IPv6 IGP-Prefix Segment ID Sub-TLV

The Sub-TLV format for IPv6 IGP-Prefix Segment ID MUST be set as shown in the below TLV format:



MT-ID identifies the Multi-Topology ID associated with the Prefix SID. MT-ID is set in trailing 12 bits of the field when the Protocol is set to IS-IS. Leading 4-bits of the MT-ID MUST be all zeroes for IS-IS. MT-ID is trailing 8 bits when the protocol is specified as OSPF. The leading octet MUST be set to all zeroes for OSPF. MBZ MUST be set to all zeroes.

The Protocol field MUST be set 1 if the responder MUST perform FEC validation using OSPF as the IGP protocol and MT-ID is an OSPF Multi-Topology ID. Protocol is set to 2 if the responder MUST perform FEC validation using IS-IS as the IGP protocol, and the MT-ID is IS-IS Multi-Topology ID. Protocol MUST not be set to 0 when using Multi-Topology IPv6 IGP Prefix SID sub-TLV.

6. Procedures

The below section describes LSP Ping and Traceroute procedures beyond the text specified in LSP

6.1. Single-Topology Networks

An array of network operators may deploy flexible algorithms in their network for constraint-based shortest paths, without deploying multi-topology. The updated FEC definitions for IGP Prefix SID allows operator to achieve LSP Ping and Traceroute in these networks while maintaining backwards compatibility with existing devices in the network. Below text highlights the handling procedures and initiator and responder for the updated FEC definitions.

6.1.1. Initiator Node Procedures

A node initiating LSP echo request packet for the Node Segment ID MUST identify and include the algorithm associated with the IGP Prefix SID in the Target FEC Stack sub-TLV. If the initiating node

is not aware of the algorithm, the default algorithm (id 0) of Shortest Path First is assumed.

6.1.2. Responder Node Procedures

This section updates the procedures defined in Section 7.4 of [RFC8287] for IPv4/IPv6 IGP Prefix SID FEC. If the algorithm is 0, the procedures from [RFC8287] do not require any change. For any other algorithm value, if the responding node is validating the FEC stack, it MUST also validate the IGP Prefix SID advertisement for the algorithm defined in Algo field.

If the responding node is including IGP Prefix SID FEC in the FEC stack due to FEC Stack Change operation, it MUST also include algorithm associated with the Prefix SID.

6.2. Multi-Topology Networks

In presence of Multi-Topology networks, the operators can use the new Multi-Topology IGP IPv4/IPv6 Prefix SID FEC definitions to achieve path validation and fault isolation. Below text describes handling procedures for Multi-Topology networks for initiator and responder. The procedures defined in [RFC8287] are still applicable and the text below updates them instead of replacing them.

6.2.1. Initiator Node Procedures

A node initiating LSP echo request packet for Single-Topology network MAY use Multi-Topology IGP IPv4/IPv6 Prefix SID defined above. A node initiating LSP echo request for Multi-Topology networks MUST use Multi-Topology IGP IPv4/IPv6 Prefix SID defined above. The node MUST identify and include both the IGP MT-ID and the algorithm associated with the IGP prefix SID in addition to prefix, prefix length, and the protocol. If the initiating node is not aware of the algorithm, the default algorithm (id 0) of Shortest Path First is assumed. The protocol MUST be set to 1 if the responding node is running OSPF, and 2 if the responding node is running IS-IS.

6.2.2. Responding Node Procedures

This section updates the procedures defined in Section 7.4 of [RFC8287] for Multi-Topology IPv4/IPv6 IGP Prefix SID FEC. Upon reception of the sub-TLV, responding node MUST validate that Protocol field is not 0 to correctly parse MT-ID. In addition to procedures defined in [RFC8287], if responding node is validating the FEC Stack, it MUST validate the IGP Prefix SID advertisement for the algorithm and the MT-ID described in the incoming FEC sub-TLV.

If the responding node is including Multi-Topology IGP Prefix SID FEC in the FEC stack due to a FEC Stack Change operation, it MUST also include the algorithm and MT-ID associated with the Prefix SID, and set the Protocol to 1 or 2, based on the corresponding IGP.

7. IANA Considerations

TBD.

8. Security Considerations

This document updates [RFC8287] and does not introduce any security considerations.

9. Acknowledgements

TBA.

10. Contributors

TBA

11. References

11.1. Normative References

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

[RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <https://www.rfc-editor.org/info/rfc8029>.

[RFC8287] Kumar, N., Ed., Pignataro, C., Ed., Swallow, G., Akiya, N., Kini, S., and M. Chen, "Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes", RFC 8287, DOI 10.17487/RFC8287, December 2017, <https://www.rfc-editor.org/info/rfc8287>.

11.2. Informative References

[I-D.ietf-lsr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-algo-00 (work in progress), May 2018.

[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-14 (work in progress), June 2018.

[RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, DOI 10.17487/RFC4915, June 2007, <https://www.rfc-editor.org/info/rfc4915>.

[RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <https://www.rfc-editor.org/info/rfc5120>.

[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>.

Authors' Addresses

Faisal Iqbal (editor)
Cisco Systems, Inc.

Email: faiqbal@cisco.com

Nagendra Kumar
Cisco Systems, Inc.

Email: naikumar@cisco.com

Zafar Ali
Cisco Systems, Inc.

Email: zali@cisco.com

Carlos Pignataro
Cisco Systems, Inc.

Email: cpignata@cisco.com

SPRING
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

F. Clad, Ed.
Cisco Systems, Inc.
X. Xu, Ed.
Alibaba
C. Filsfils
Cisco Systems, Inc.
D. Bernier
Bell Canada
C. Li
Huawei
B. Decraene
Orange
S. Ma
Juniper
C. Yadlapalli
AT&T
W. Henderickx
Nokia
S. Salsano
Universita di Roma "Tor Vergata"
July 02, 2018

Service Programming with Segment Routing
draft-xuclad-spring-sr-service-programming-00

Abstract

This document defines data plane functionality required to implement service segments and achieve service programming in SR-enabled MPLS and IP networks, as described in the Segment Routing architecture.

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 3, 2019.

Table of Contents

1. Introduction

Segment Routing (SR) is an architecture based on the source routing paradigm that seeks the right balance between distributed intelligence and centralized programmability. SR can be used with an MPLS or an IPv6 data plane to steer packets through an ordered list of instructions, called segments. These segments may encode simple routing instructions for forwarding packets along a specific network path, but also steer them through VNFs or physical service appliances available in the network.

In an SR network, each of these services, running either on a physical appliance or in a virtual environment, are associated with a segment identifier (SID). These service SIDs are then leveraged as part of a SID-list to steer packets through the corresponding services. Service SIDs may be combined together in a SID-list to achieve service programming, but also with other types of segments as defined in [I-D.ietf-spring-segment-routing]. SR thus provides a fully integrated solution for overlay, underlay and service programming. Furthermore, the IPv6 instantiation of SR (SRv6) supports metadata transportation in the Segment Routing header [I-D.ietf-6man-segment-routing-header], either natively in the tag field or with extensions such as TLVs.

This document describes how a service can be associated with a SID, including legacy services with no SR capabilities, and how these service SIDs are integrated within an SR policy. The definition of an SR Policy and the traffic steering mechanisms are covered in [I-D.ietf-spring-segment-routing-policy] and hence outside the scope of this document.

The definition of control plane components, such as service segment discovery, is outside the scope of this data plane document. For reference, the option of using BGP extensions to support SR service programming is proposed in [I-D.dawra-idr-bgp-sr-service-chaining].

2. Terminology

This document leverages the terminology proposed in [I-D.ietf-spring-segment-routing] and [I-D.ietf-spring-segment-routing-policy]. It also introduces the following new terms.

Service segment: A segment associated with a service. The service may either run on a physical appliance or in a virtual environment such as a virtual machine or container.

SR-aware service: A service that is fully capable of processing SR traffic. An SR-aware service can be directly associated with a service segment.

SR-unaware service: A service that is unable to process SR traffic or may behave incorrectly due to presence of SR information in the packet headers. An SR-unaware service can be associated with a service segment through an SR proxy function.

3. Classification and steering

Classification and steering mechanisms are defined in section 8 of [I-D.ietf-spring-segment-routing-policy] and are independent from the purpose of the SR policy. From the perspective of a headend node classifying and steering traffic into an SR policy, there is no difference whether this policy contains IGP, BGP, peering, VPN or service segments, or any combination of these.

As documented in the above reference, traffic is classified when entering an SR domain. The SR policy headend may, depending on its capabilities, classify the packets on a per-destination basis, via simple FIB entries, or apply more complex policy routing rules requiring to look deeper into the packet. These rules are expected to support basic policy routing such as 5-tuple matching. In addition, the IPv6 SRH tag field defined in [I-D.ietf-6man-segment-routing-header] can be used to identify and classify packets sharing the same set of properties. Classified traffic is then steered into the appropriate SR policy and forwarded as per the SID-list(s) of the active candidate path.

SR traffic can be re-classified by an SR endpoint along the original SR policy (e.g., DPI service) or a transit node intercepting the traffic. This node is the head-end of a new SR policy that is imposed onto the packet, either as a stack of MPLS labels or as an IPv6 SRH.

4. Service segments

In the context of this document, the term service refers to a physical appliance running on dedicated hardware, a virtualized service inside an isolated environment such as a VM, container or namespace, or any process running on a compute element. A service may also comprise multiple sub-components running in different processes or containers. Unless otherwise stated, this document does not make any assumption on the type or execution environment of a service.

The execution of a service can be integrated as part of an SR policy by assigning a segment identifier, or SID, to the service and including this service SID in the SR policy SID-list. Such a service SID may be of local or global significance. In the former case, other segments, such as prefix or adjacency segments, can be used to steer the traffic up to the node where the service segment is instantiated. In the latter case, the service is directly reachable from anywhere in the routing domain. This is realized with SR-MPLS by assigning a SID from the global label block ([I-D.ietf-spring-segment-routing-mpls]), or with SRv6 by advertising the SID locator in the routing protocol ([I-D.filsfils-spring-srv6-network-programming]). It is up to the network operator to define the scope and reachability of each service SID. This decision can be based on various considerations such as infrastructure dynamicity, available control plane or orchestration system capabilities.

This document categorizes services in two types, depending on whether they are able to behave properly in the presence of SR information or not. These are respectively named SR-aware and SR-unaware services.

4.1. SR-aware services

An SR-aware service can process the SR information in the packets it receives. This means being able to identify the active segment as a local instruction and move forward in the segment list, but also that the service’s own behavior is not hindered due to the presence of SR information. For example, an SR-aware firewall filtering SRv6 traffic based on its final destination must retrieve that information from the last entry in the SRH rather than the Destination Address field of the IPv6 header.

An SR-aware service is associated with a locally instantiated service segment, which is used to steer traffic through it.

If the service is configured to intercept all the packets passing through the appliance, the underlying routing system only has to

implement a default SR endpoint behavior (SR-MPLS node segment or SRv6 End function), and the corresponding SID will be used to steer traffic through the service.

If the service requires the packets to be directed to a specific virtual interface, networking queue or process, a dedicated SR behavior may be required to steer the packets to the appropriate location. The definition of such service-specific functions is out of the scope of this document.

SR-aware services also enable advanced network programming functionalities such as conditional branching and jumping to arbitrary SIDs in the segment list. In addition, SRv6 provides several ways of passing and exchanging information between services (e.g., SID arguments, tag field and TLVs). An example scenario involving these features is described in [IFIP18], which discusses the implementation of an SR-aware Intrusion Detection System.

Examples of SR-aware services are provided in section Section 8.1.

4.2. SR-unaware services

Any service that does not meet the above criteria for SR-awareness is considered as SR-unaware.

An SR-unaware service is not able to process the SR information in the traffic that it receives. It may either drop the traffic or take erroneous decisions due to the unrecognized routing information. In order to include such services in an SR policy, it is thus required to remove the SR information as well as any other encapsulation header before the service receives the packet, or to alter it in such a way that the service can correctly process the packet.

In this document, we define the concept of an SR proxy as an entity, separate from the service, that performs these modifications and handle the SR processing on behalf of a service. The SR proxy can run as a separate process on the service appliance, on a virtual switch or router on the compute node or on a different host.

An SR-unaware service is associated with a service segment instantiated on the SR proxy, which is used to steer traffic through the service. Section 6 describes several SR proxy behaviors to handle the encapsulation headers and SR information under various circumstances.

5. SR service policies

An SR service policy is an SR policy, as defined in [I-D.ietf-spring-segment-routing-policy], that includes at least one service. This service is represented in the SID-list by its associated service SID. In case the policy should include several services, the service traversal order is indicated by the relative position of each service SID in the SID-list. Using the mechanisms described in [I-D.ietf-spring-segment-routing-policy], it is possible to load balance the traffic over several services, or instances of the same service, by associating with the SR service policy a weighted set of SID-lists, each containing a possible sequence of service SIDs to be traversed. Similarly, several candidate paths can be specified for the SR service policy, each with its own set of SID-lists, for resiliency purposes.

Furthermore, binding SIDs (BSIDs) can be leveraged in the context of service policies to reduce the number of SIDs imposed by the headend, provide opacity between domains and improve scalability, as described in [I-D.filsfils-spring-sr-policy-considerations]. For example, a network operator may want a policy in its core domain to include services that are running in one of its datacenters. One option is to define an SR policy at ingress edge of the core domain that explicitly includes all the SIDs needed to steer the traffic through the core and in the DC, but that may result in a long SID-list and requires to update the ingress edge configuration every time the DC part of the policy is modified. Alternatively, a separate policy can be defined at the ingress edge of the datacenter with only the SIDs that needs to be executed there and its BSID included in the core domain policy. That BSID remains stable when the DC policy is modified and can even be shared among several core domain policies that would require the same type of processing in the DC.

This section describes how services can be integrated within an SR-MPLS or SRv6 service policy.

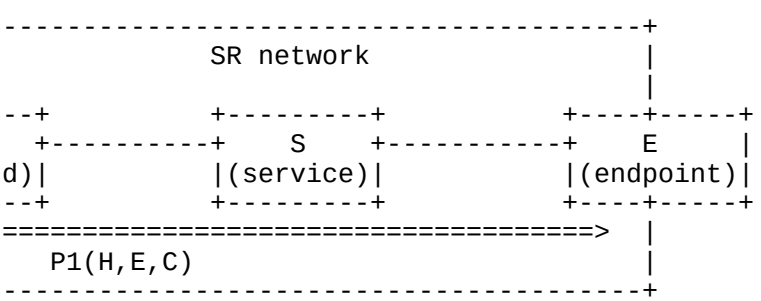


Figure 1: SR service policy

Figure 1 illustrates a basic SR service policy instantiated on a headend node H towards an endpoint E and traversing a service S. The SR policy may also include additional requirements, such as traffic engineering or VPN. On the head-end H, the SR policy P1 is created with a color C and endpoint E and associated with an SR path that can either be explicitly configured, dynamically computed on H or provisioned by a network controller.

In its most basic form, the SR policy P1 would be resolved into the SID-list < SID(S), SID(E) >. This is assuming that SID(S) and SID(E) are directly reachable from H and S, respectively, and that the forwarding path meets the policy requirement. However, depending on the dataplane and the segments available in the network, additional SIDs may be required to enforce the SR policy.

This model applies regardless of the SR-awareness of the service. If it is SR-unaware, then S simply represents the proxy that takes care of transmitting the packet to the actual service.

Traffic can then be steered into this policy using any of the mechanisms described in [I-D.ietf-spring-segment-routing-policy].

The following subsections describe the specificities of each SR dataplane.

5.1. SR-MPLS data plane

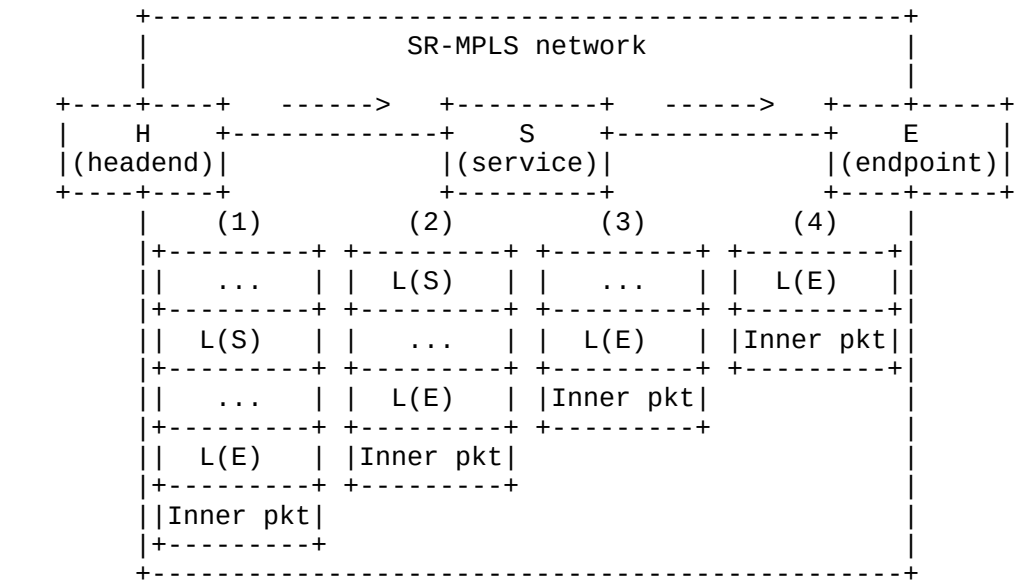


Figure 2: Packet walk in an SR-MPLS network

In an SR-MPLS network, the SR policy SID-list is encoded as a stack of MPLS labels[I-D.ietf-spring-segment-routing-mpls] and pushed on top of the packet.

In the example shown on Figure 2, the SR policy should steer the traffic from the head-end H to the endpoint E via a service S. This translates into an MPLS label stack that includes at least a label L(S) associated to service S and a label L(E) associated to the endpoint E. The label stack may also include additional intermediate segments if these are required for traffic engineering (e.g., to encode a low latency path between H and S and / or between S and E) or simply for reachability purposes. Indeed, the service SID L(S) may be taken from the global or local SID block of node S and, in the latter case, one or more SIDs might be needed before L(S) in order for the packet to reach node S (e.g., a prefix-SID of S), where L(S) can be interpreted. The same applies for the segment L(E) at the SR policy endpoint.

Special consideration must be taken into account when using Local SIDs for service identification due to increased label stack depth and the associated impacts.

When the packet arrives at S, this node determines how to process the packet based on the semantic locally associated to the top label L(S). If S is an SR-aware service, the SID L(S) may provide

additional context or indication on how to process the packet (e.g., payload type or a firewall SID may indicate which rule set should be applied onto the packet). If S is a proxy in front of an SR-unaware service, L(S) indicates how and to which service attached to this proxy the packet should be transmitted. At some point in the process, L(S) is also popped from the label stack in order to expose the next SID, which may be L(E) or another intermediate segment.

5.2. SRv6 data plane

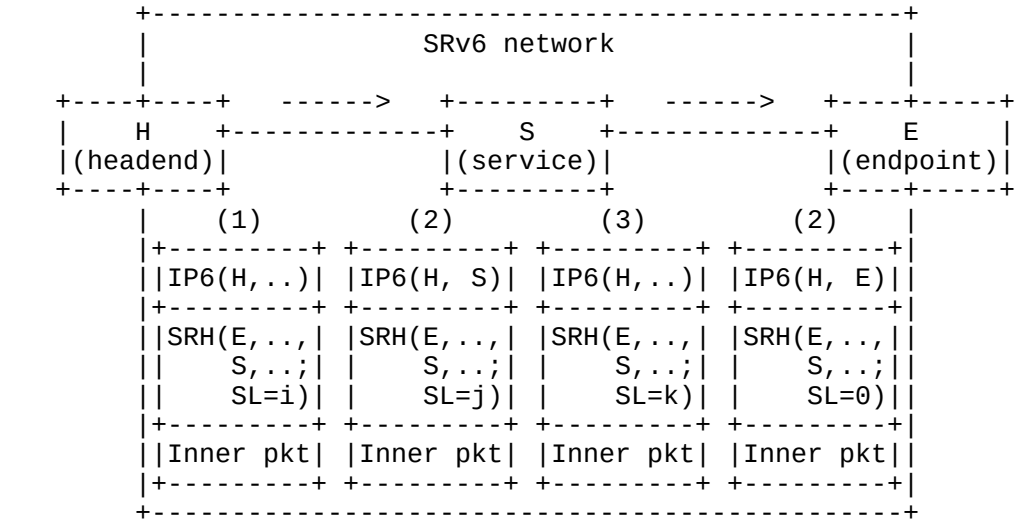


Figure 3: Packet walk in an SRv6 network

In an SRv6 network, the SR Policy is encoded into the packet as an IPv6 header possibly followed by a Segment Routing header (SRH) [I-D.ietf-6man-segment-routing-header].

In the example shown on Figure 3, the SR policy should steer the traffic from the head-end H to the endpoint E via a service S. This translates into an SRH that includes at least a segment SID(S) to the service, or service proxy, S and a segment SID(E) to the endpoint E. The SRH may also include additional intermediate segments if these are required for traffic engineering (e.g., the encode a low latency path between H and S and / or between S and E) or simply for reachability purposes. Indeed, the service segment locator may or may not be advertised in the routing protocol and, in the latter case, one or more SIDs might be needed before SID(S) in order to bring the packet up to node S, where SID(S) can be interpreted. The same applies for the segment SID(E) at the SR policy endpoint.

When the packet arrives at S, this node determines how to process the packet based on the semantic locally associated to the active segment SID(S). If S is an SR-aware service, then SID(S) may provide additional context or indication on how to process the packet (e.g., a firewall SID may indicate which rule set should be applied onto the packet). If S is a proxy in front of an SR-unaware service, SID(S) indicates how and to which service attached to this proxy the packet should be transmitted. At some point in the process, the SRv6 End function is also applied in order to make the next SID, which may be SID(E) or another intermediate segment, active.

The "Inner pkt" on Figure 3 represents the SRv6 payload, which may be an encapsulated IP packet, an Ethernet frame or a transport-layer payload, for example.

6. SR proxy behaviors

This section describes several SR proxy behaviors designed to enable SR service programming through SR-unaware services. A system implementing one of these functions may handle the SR processing on behalf of an SR-unaware service and allows the service to properly process the traffic that is steered through it.

A service may be located at any hop in an SR policy, including the last segment. However, the SR proxy behaviors defined in this section are dedicated to supporting SR-unaware services at intermediate hops in the segment list. In case an SR-unaware service is at the last segment, it is sufficient to ensure that the SR information is ignored (IPv6 routing extension header with Segments Left equal to 0) or removed before the packet reaches the service (MPLS PHP, SRv6 End.D or PSP).

As illustrated on Figure 4, the generic behavior of an SR proxy has two parts. The first part is in charge of passing traffic from the network to the service. It intercepts the SR traffic destined for the service via a locally instantiated service segment, modifies it in such a way that it appears as non-SR traffic to the service, then sends it out on a given interface, IFACE-OUT, connected to the service. The second part receives the traffic coming back from the service on IFACE-IN, restores the SR information and forwards it according to the next segment in the list. IFACE-OUT and IFACE-IN are respectively the proxy interface used for sending traffic to the service and the proxy interface that receives the traffic coming back from the service. These can be physical interfaces or sub-interfaces (VLANs) and, unless otherwise stated, IFACE-OUT and IFACE-IN can represent the same interface.

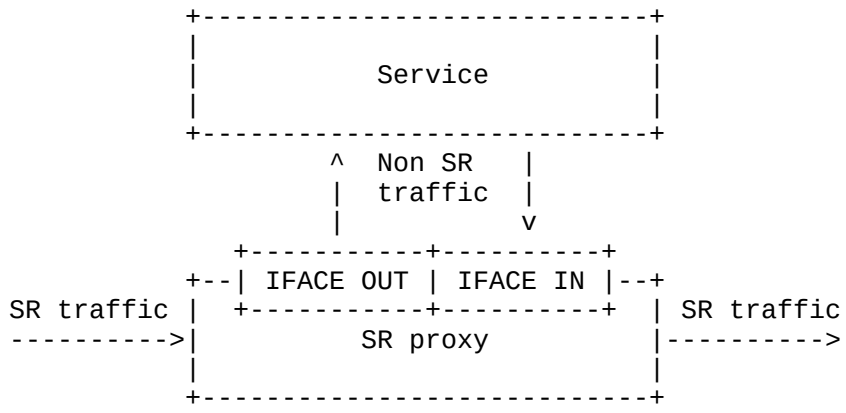


Figure 4: Generic SR proxy

In the next subsections, the following SR proxy mechanisms are defined:

- Static proxy
- Dynamic proxy
- Shared-memory proxy
- Masquerading proxy

Each mechanism has its own characteristics and constraints, which are summarized in the below table. It is up to the operator to select the best one based on the proxy node capabilities, the service behavior and the traffic type. It is also possible to use different proxy mechanisms within the same service policy.

		Static	Dynamic	Shared mem.	Masquerading
SR flavors	SR-MPLS	Y	Y	Y	-
	SRv6 insertion	P	P	P	Y
	SRv6 encapsulation	Y	Y	Y	-
Chain agnostic configuration		N	N	Y	Y
Transparent to chain changes		N	Y	Y	Y
Service support	DA modification	Y	Y	Y	NAT
	Payload modification	Y	Y	Y	Y
	Packet generation	Y	Y	cache	cache
	Packet deletion	Y	Y	Y	Y
	Transport endpoint	Y	Y	cache	cache
Supported traffic	Ethernet	Y	Y	Y	-
	IPv4	Y	Y	Y	-
	IPv6	Y	Y	Y	Y

Figure 5: SR proxy summary

Note: The use of a shared memory proxy requires both the service (VNF) and the proxy to be running on the same node.

6.1. Static SR proxy

The static proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware service. This proxy thus receives SR traffic that is formed of an MPLS label stack or an IPv6 header on top of an inner packet, which can be Ethernet, IPv4 or IPv6.

A static SR proxy segment is associated with the following mandatory parameters

- INNER-TYPE: Inner packet type
- NH-ADDR: Next hop Ethernet address (only for inner type IPv4 and IPv6)
- IFACE-OUT: Local interface for sending traffic towards the service
- IFACE-IN: Local interface receiving the traffic coming back from the service
- CACHE: SR information to be attached on the traffic coming back from the service, including at least
  - CACHE.SA: IPv6 source address (SRv6 only)
  - CACHE.LIST: Segment list expressed as MPLS labels or IPv6 address

A static SR proxy segment is thus defined for a specific service, inner packet type and cached SR information. It is also bound to a pair of directed interfaces on the proxy. These may be both directions of a single interface, or opposite directions of two different interfaces. The latter is recommended in case the service is to be used as part of a bi-directional SR SC policy. If the proxy and the service both support 802.1Q, IFACE-OUT and IFACE-IN can also represent sub-interfaces.

The first part of this behavior is triggered when the proxy node receives a packet whose active segment matches a segment associated with the static proxy behavior. It removes the SR information from the packet then sends it on a specific interface towards the associated service. This SR information corresponds to the full label stack for SR-MPLS or to the encapsulation IPv6 header with any attached extension header in the case of SRv6.

The second part is an inbound policy attached to the proxy interface receiving the traffic returning from the service, IFACE-IN. This

policy attaches to the incoming traffic the cached SR information associated with the SR proxy segment. If the proxy segment uses the SR-MPLS data plane, CACHE contains a stack of labels to be pushed on top of the packets. With the SRv6 data plane, CACHE is defined as a source address, an active segment and an optional SRH (tag, segments left, segment list and metadata). The proxy encapsulates the packets with an IPv6 header that has the source address, the active segment as destination address and the SRH as a routing extension header. After the SR information has been attached, the packets are forwarded according to the active segment, which is represented by the top MPLS label or the IPv6 Destination Address. An MPLS TTL or IPv6 Hop Limit value may also be configured in CACHE. If it is not, the proxy should set these values according to the node's default setting for MPLS or IPv6 encapsulation.

In this scenario, there are no restrictions on the operations that can be performed by the service on the stream of packets. It may operate at all protocol layers, terminate transport layer connections, generate new packets and initiate transport layer connections. This behavior may also be used to integrate an IPv4-only service into an SRv6 policy. However, a static SR proxy segment can be used in only one service policy at a time. As opposed to most other segment types, a static SR proxy segment is bound to a unique list of segments, which represents a directed SR SC policy. This is due to the cached SR information being defined in the segment configuration. This limitation only prevents multiple segment lists from using the same static SR proxy segment at the same time, but a single segment list can be shared by any number of traffic flows. Besides, since the returning traffic from the service is re-classified based on the incoming interface, an interface can be used as receiving interface (IFACE-IN) only for a single SR proxy segment at a time. In the case of a bi-directional SR SC policy, a different SR proxy segment and receiving interface are required for the return direction.

The static proxy behavior may also be used for sending traffic through "bump in the wire" services that are transparent to the IP and Ethernet layers. This type of processing is assumed when the inner traffic type is Ethernet, since the original destination address of the Ethernet frame is preserved when the packet is steered into the SR Policy and likely associated with a node downstream of the policy tail-end. In case the inner type is IP (IPv4 or IPv6), the NH-ADDR parameter may be set to a dummy or broadcast Ethernet address, or simply to the address of the proxy receiving interface (IFACE-IN).

6.1.1. SR-MPLS pseudocode

6.1.1.1. Static proxy for inner type Ethernet

Upon receiving an MPLS packet with top label L, where L is an MPLS L2 static proxy segment, a node N does:

```
1.  Pop all labels
2.  IF payload type is Ethernet THEN
3.      Forward the exposed frame on IFACE-OUT
4.  ELSE
5.      Drop the packet
```

Upon receiving on IFACE-IN an Ethernet frame with a destination address different than the interface address, a node N does:

```
1.  Push labels in CACHE on top of the frame Ethernet header
2.  Lookup the top label and proceed accordingly
```

The receiving interface must be configured in promiscuous mode in order to accept those Ethernet frames.

6.1.1.2. Static proxy for inner type IPv4

Upon receiving an MPLS packet with top label L, where L is an MPLS IPv4 static proxy segment, a node N does:

```
1.  Pop all labels
2.  IF payload type is IPv4 THEN
3.      Forward the exposed packet on IFACE-OUT towards NH-ADDR
4.  ELSE
5.      Drop the packet
```

Upon receiving a non-link-local IPv4 packet on IFACE-IN, a node N does:

```
1.  Decrement TTL and update checksum
2.  Push labels in CACHE on top of the packet IPv4 header
3.  Lookup the top label and proceed accordingly
```

6.1.1.3. Static proxy for inner type IPv6

Upon receiving an MPLS packet with top label L, where L is an MPLS IPv6 static proxy segment, a node N does:

```
1.   Pop all labels
2.   IF payload type is IPv6 THEN
3.       Forward the exposed packet on IFACE-OUT towards NH-ADDR
4.   ELSE
5.       Drop the packet
```

Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N does:

```
1.   Decrement Hop Limit
2.   Push labels in CACHE on top of the packet IPv6 header
3.   Lookup the top label and proceed accordingly
```

6.1.2. SRv6 pseudocode

6.1.2.1. Static proxy for inner type Ethernet

Upon receiving an IPv6 packet destined for S, where S is an IPv6 static proxy segment for Ethernet traffic, a node N does:

```
1.   IF ENH == 59 THEN                                          ;; Ref1
2.       Remove the (outer) IPv6 header and its extension headers
3.       Forward the exposed frame on IFACE-OUT
4.   ELSE
5.       Drop the packet
```

Ref1: 59 refers to "no next header" as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving on IFACE-IN an Ethernet frame with a destination address different than the interface address, a node N does:

```
1.   Retrieve CACHE entry matching IFACE-IN and traffic type
2.   Push SRH with CACHE.LIST on top of the Ethernet header   ;; Ref2
3.   Push IPv6 header with
       SA = CACHE.SA
       DA = CACHE.LIST[0]                                     ;; Ref3
       Next Header = 43                                       ;; Ref4
4.   Set outer payload length and flow label
5.   Lookup outer DA in appropriate table and proceed accordingly
```

Ref2: Unless otherwise specified, the segments in CACHE.LIST should be encoded in reversed order, Segment Left and Last Entry values should be set of the length of CACHE.LIST minus 1, and Next Header should be set to 59.

Ref3: CACHE.LIST[0] represents the first IPv6 SID in CACHE.LIST.

Ref4: If CACHE.LIST contains a single entry, the SRH can be omitted and the Next Header value must be set to 59.

The receiving interface must be configured in promiscuous mode in order to accept those Ethernet frames.

6.1.2.2. Static proxy for inner type IPv4

Upon receiving an IPv6 packet destined for S, where S is an IPv6 static proxy segment for IPv4 traffic, a node N does:

```
1.   IF ENH == 4 THEN                                          ;; Ref1
2.       Remove the (outer) IPv6 header and its extension headers
3.       Forward the exposed packet on IFACE-OUT towards NH-ADDR
4.   ELSE
5.       Drop the packet
```

Ref1: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving a non-link-local IPv4 packet on IFACE-IN, a node N does:

```
1.   Decrement TTL and update checksum
2.   IF CACHE.SRH THEN                                          ;; Ref2
3.       Push CACHE.SRH on top of the existing IPv4 header
4.       Set NH value of the pushed SRH to 4
5.   Push outer IPv6 header with SA, DA and traffic class from CACHE
6.   Set outer payload length and flow label
7.   Set NH value to 43 if an SRH was added, or 4 otherwise
8.   Lookup outer DA in appropriate table and proceed accordingly
```

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

6.1.2.3. Static proxy for inner type IPv6

Upon receiving an IPv6 packet destined for S, where S is an IPv6 static proxy segment for IPv6 traffic, a node N does:

```
1.   IF ENH == 41 THEN                                         ;; Ref1
2.       Remove the (outer) IPv6 header and its extension headers
3.       Forward the exposed packet on IFACE-OUT towards NH-ADDR
4.   ELSE
5.       Drop the packet
```

Ref1: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N does:

```
1.  Decrement Hop Limit
2.  IF CACHE.SRH THEN                                        ;; Ref2
3.      Push CACHE.SRH on top of the existing IPv6 header
4.      Set NH value of the pushed SRH to 41
5.  Push outer IPv6 header with SA, DA and traffic class from CACHE
6.  Set outer payload length and flow label
7.  Set NH value to 43 if an SRH was added, or 41 otherwise
8.  Lookup outer DA in appropriate table and proceed accordingly
```

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

6.2. Dynamic SR proxy

The dynamic proxy is an improvement over the static proxy that dynamically learns the SR information before removing it from the incoming traffic. The same information can then be re-attached to the traffic returning from the service. As opposed to the static SR proxy, no CACHE information needs to be configured. Instead, the dynamic SR proxy relies on a local caching mechanism on the node instantiating this segment.

Upon receiving a packet whose active segment matches a dynamic SR proxy function, the proxy node pops the top MPLS label or applies the SRv6 End behavior, then compares the updated SR information with the cache entry for the current segment. If the cache is empty or different, it is updated with the new SR information. The SR information is then removed and the inner packet is sent towards the service.

The cache entry is not mapped to any particular packet, but instead to an SR SC policy identified by the receiving interface (IFACE-IN). Any non-link-local IP packet or non-local Ethernet frame received on that interface will be re-encapsulated with the cached headers as described in Section 6.1. The service may thus drop, modify or generate new packets without affecting the proxy.

6.2.1. SR-MPLS pseudocode

The dynamic proxy SR-MPLS pseudocode is obtained by inserting the following instructions at the beginning of the static SR-MPLS pseudocode (Section 6.1.1).

```
1.   IF top label S bit is 0 THEN                              ;; Ref1
2.       Pop top label
3.       IF C(IFACE-IN) different from remaining labels THEN  ;; Ref2
4.           Copy all remaining labels into C(IFACE-IN)       ;; Ref3
5.   ELSE
6.       Drop the packet
```

Ref1: As mentioned at the beginning of Section 6, an SR proxy is not needed to include an SR-unaware service at the end of an SR policy.

Ref2: A TTL margin can be configured for the top label stack entry to prevent constant cache updates when multiple equal-cost paths with different hop counts are used towards the SR proxy node. In that case, a TTL difference smaller than the configured margin should not trigger a cache update (provided that the labels are the same).

Ref3: C(IFACE-IN) represents the cache entry associated to the dynamic SR proxy segment. It is identified with IFACE-IN in order to efficiently retrieve the right SR information when a packet arrives on this interface.

In addition, the inbound policy should check that C(IFACE-IN) has been defined before attempting to restore the MPLS label stack and drop the packet otherwise.

6.2.2. SRv6 pseudocode

The dynamic proxy SRv6 pseudocode is obtained by inserting the following instructions between lines 1 and 2 of the static proxy SRv6 pseudocode.

```
1.   IF NH=SRH & SL > 0 THEN                                  ;; Ref1
2.       Decrement SL and update the IPv6 DA with SRH[SL]
3.       IF C(IFACE-IN) different from IPv6 encaps THEN       ;; Ref2
4.           Copy the IPv6 encaps into C(IFACE-IN)            ;; Ref3
5.   ELSE
6.       Drop the packet
```

Ref1: As mentioned at the beginning of Section 6, an SR proxy is not needed to include an SR-unaware service at the end of an SR policy.

Ref2: "IPv6 encaps" represents the IPv6 header and any attached extension header.

Ref3: C(IFACE-IN) represents the cache entry associated to the dynamic SR proxy segment. It is identified with IFACE-IN in order to efficiently retrieve the right SR information when a packet arrives on this interface.

In addition, the inbound policy should check that C(IFACE-IN) has been defined before attempting to restore the IPv6 encapsulation and drop the packet otherwise.

6.3. Shared memory SR proxy

The shared memory proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware service. This proxy behavior leverages a shared-memory interface with a virtualized service (VNF) in order to hide the SR information from an SR-unaware service while keeping it attached to the packet. We assume in this case that the proxy and the VNF are running on the same compute node. A typical scenario is an SR-capable vrouter running on a container host and forwarding traffic to VNFs isolated within their respective container.

More details will be added in a future revision of this document.

6.4. Masquerading SR proxy

The masquerading proxy is an SR endpoint behavior for processing SRv6 traffic on behalf of an SR-unaware service. This proxy thus receives SR traffic that is formed of an IPv6 header and an SRH on top of an inner payload. The masquerading behavior is independent from the inner payload type. Hence, the inner payload can be of any type but it is usually expected to be a transport layer packet, such as TCP or UDP.

A masquerading SR proxy segment is associated with the following mandatory parameters:

- S-ADDR: Ethernet or IPv6 address of the service
- IFACE-OUT: Local interface for sending traffic towards the service
- IFACE-IN: Local interface receiving the traffic coming back from the service

A masquerading SR proxy segment is thus defined for a specific service and bound to a pair of directed interfaces or sub-interfaces on the proxy. As opposed to the static and dynamic SR proxies, a masquerading segment can be present at the same time in any number of SR SC policies and the same interfaces can be bound to multiple masquerading proxy segments. The only restriction is that a masquerading proxy segment cannot be the last segment in an SR SC policy.

The first part of the masquerading behavior is triggered when the proxy node receives an IPv6 packet whose Destination Address matches a masquerading proxy segment. The proxy inspects the IPv6 extension headers and substitutes the Destination Address with the last segment in the SRH attached to the IPv6 header, which represents the final destination of the IPv6 packet. The packet is then sent out towards the service.

The service receives an IPv6 packet whose source and destination addresses are respectively the original source and final destination. It does not attempt to inspect the SRH, as RFC8200 specifies that routing extension headers are not examined or processed by transit nodes. Instead, the service simply forwards the packet based on its current Destination Address. In this scenario, we assume that the service can only inspect, drop or perform limited changes to the packets. For example, Intrusion Detection Systems, Deep Packet Inspectors and non-NAT Firewalls are among the services that can be supported by a masquerading SR proxy. Variants of the masquerading behavior are defined in Section 6.4.2 and Section 6.4.3 to support a wider range of services.

The second part of the masquerading behavior, also called de-masquerading, is an inbound policy attached to the proxy interface receiving the traffic returning from the service, IFACE-IN. This policy inspects the incoming traffic and triggers a regular SRv6 endpoint processing (End) on any IPv6 packet that contains an SRH. This processing occurs before any lookup on the packet Destination Address is performed and it is sufficient to restore the right active segment as the Destination Address of the IPv6 packet.

6.4.1. SRv6 masquerading proxy pseudocode

Masquerading: Upon receiving a packet destined for S, where S is an IPv6 masquerading proxy segment, a node N processes it as follows.

```
1.   IF NH=SRH & SL > 0 THEN
2.       Update the IPv6 DA with SRH[0]
3.       Forward the packet on IFACE-OUT
4.   ELSE
5.       Drop the packet
```

De-masquerading: Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N processes it as follows.

```
1.   IF NH=SRH & SL > 0 THEN
2.       Decrement SL
3.       Update the IPv6 DA with SRH[SL]                    ;; Ref1
4.       Lookup DA in appropriate table and proceed accordingly
```

Ref1: This pseudocode can be augmented to support the Penultimate Segment Popping (PSP) endpoint flavor. The exact pseudocode modification are provided in [I-D.filsfils-spring-srv6-network-programming].

6.4.2. Variant 1: Destination NAT

Services modifying the destination address in the packets they process, such as NATs, can be supported by a masquerading proxy with the following modification to the de-masquerading pseudocode.

De-masquerading - NAT: Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N processes it as follows.

```
1.   IF NH=SRH & SL > 0 THEN
2.       Update SRH[0] with the IPv6 DA
3.       Decrement SL
4.       Update the IPv6 DA with SRH[SL]
5.       Lookup DA in appropriate table and proceed accordingly
```

6.4.3. Variant 2: Caching

Services generating packets or acting as endpoints for transport connections can be supported by adding a dynamic caching mechanism similar to the one described in Section 6.2.

More details will be added in a future revision of this document.

7. Metadata

7.1. MPLS data plane

Metadata can be carried for SR-MPLS traffic in a Segment Routing header inserted between the last MPLS label and the MPLS payload. When used solely as a metadata container, the SRH does not carry any segment but only the mandatory header fields, including the tag and flags, and any TLVs that is required for transporting the metadata.

Since the MPLS encapsulation has no explicit protocol identifier field to indicate the protocol type of the MPLS payload, how to indicate the presence of metadata in an MPLS packet is a potential issue to be addressed. One possible solution is to add the indication about the presence of metadata in the semantic of the SIDs. Note that only the SIDs whose behavior involves looking at the metadata or the MPLS payload would need to include such semantic (e.g., service segments). Other segments, such as traffic engineering segments, are not affected by the presence of metadata. Another, more generic, solution is to introduce a protocol identifier

field within the MPLS packet as described in [I-D.xu-mpls-payload-protocol-identifier].

7.2. IPv6 data plane

7.2.1. SRH TLV objects

The IPv6 SRH TLV objects are designed to carry all sorts of metadata. In particular, the NSH carrier TLV is defined as a container for NSH metadata.

TLV objects can be imposed by the ingress edge router that steers the traffic into the SR SC policy.

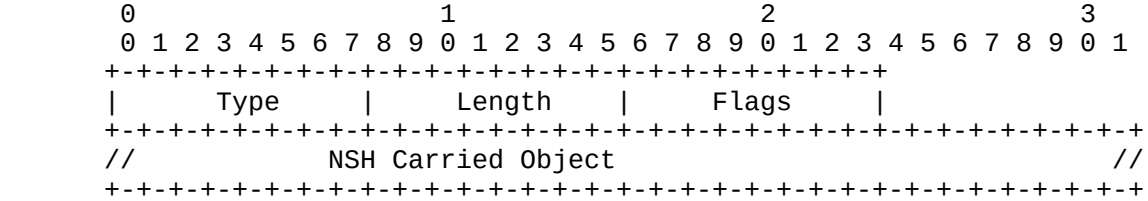
An SR-aware service may impose, modify or remove any TLV object attached to the first SRH, either by directly modifying the packet headers or via a control channel between the service and its forwarding plane.

An SR-aware service that re-classifies the traffic and steers it into a new SR SC policy (e.g. DPI) may attach any TLV object to the new SRH.

Metadata imposition and handling will be further discussed in a future version of this document.

7.2.1.1. NSH Carrier TLV

The NSH Carrier TLV is a container used in order to carry TLVs that have been defined in [RFC8300]. The NSH Carrier TLV has the following format:



where:

- Type: to be assigned by IANA (suggested value 6).
- Length: the total length of the TLV.
- Flags: 8 bits. No flags are defined in this document. SHOULD be set to 0 on transmission and MUST be ignored on receipt.

- NSH Carried Object: the content of the TLV which consists of the NSH data as defined in [RFC8300].

7.2.2. SRH tag

The SRH tag identifies a packet as part of a group or class of packets [I-D.ietf-6man-segment-routing-header].

In the context of service programming, this field can be used to encode basic metadata in the SRH. An example use case would be to leverage the SRH tag to encode a policy ID which could be leveraged in an SR-aware function to determine which processing policy to apply rather than having doing local classification or leverage alternate encapsulations.

8. Implementation status

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

8.1. SR-aware services

Specific SRv6 support has been implemented for the below open-source services:

- Iptables (1.6.2 and later)
- Nftables (0.8.4 and later)
- Snort

In addition, any service relying on the Linux kernel, version 4.10 and later, or FD.io VPP for packet forwarding can be considered as SR-aware.

8.2. Proxy behaviors

The static SR proxy is available for SR-MPLS and SRv6 on various Cisco hardware and software platforms. Furthermore, the following proxies are available on open-source software.

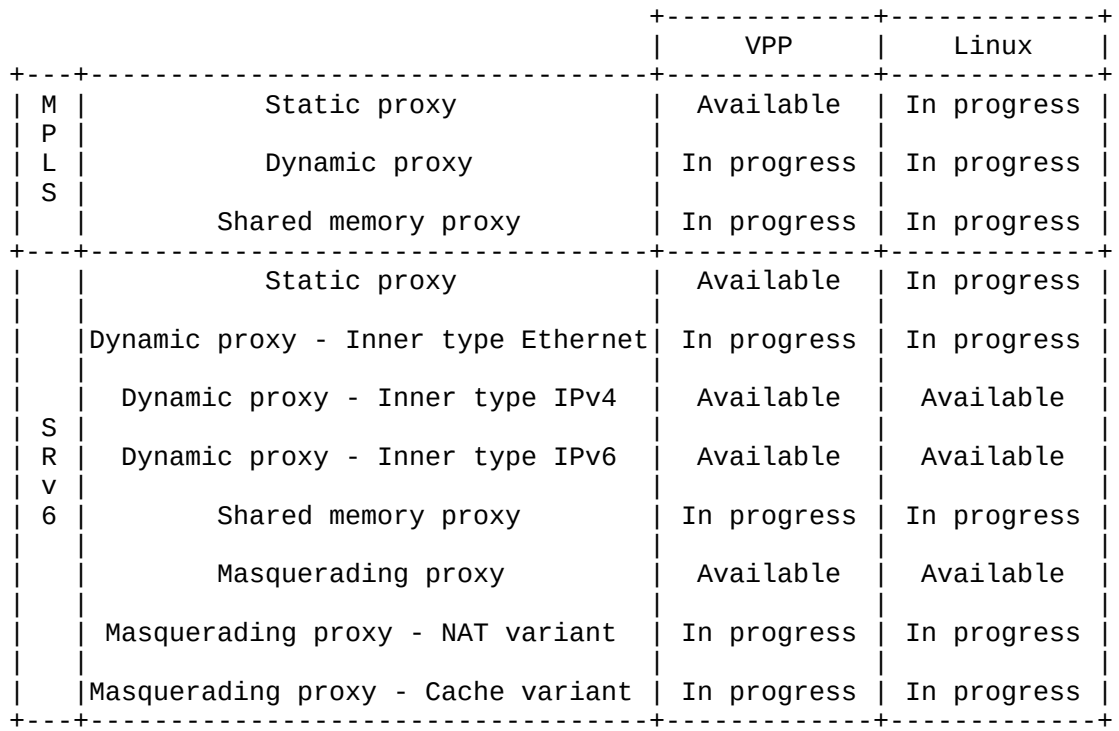


Figure 6: Open-source implementation status table

9. Related works

The Segment Routing solution addresses a wide problem that covers both topological and service policies. The topological and service instructions can be either deployed in isolation or in combination. SR has thus a wider applicability than the architecture defined in [RFC7665]. Furthermore, the inherent property of SR is a stateless network fabric. In SR, there is no state within the fabric to recognize a flow and associate it with a policy. State is only present at the ingress edge of the SR domain, where the policy is encoded into the packets. This is completely different from other proposals such as [RFC8300] and the MPLS label swapping mechanism described in [I-D.ietf-mpls-sfc], which rely on state configured at every hop of the service chain.

10. IANA Considerations

This I-D requests the IANA to allocate, within the "SRv6 Endpoint Types" sub-registry belonging to the top-level "Segment-routing with

IPv6 dataplane (SRv6) Parameters" registry, the following allocations:

Value/Range	Hex	Endpoint function	Reference
TBA	TBA	End.AN - SR-aware function (native)	[This.ID]
TBA	TBA	End.AS - Static proxy	[This.ID]
TBA	TBA	End.AD - Dynamic proxy	[This.ID]
TBA	TBA	End.AM - Masquerading proxy	[This.ID]

Figure 7: SRv6 Service Endpoint Types

11. Security Considerations

The security requirements and mechanisms described in [I-D.ietf-spring-segment-routing], [I-D.ietf-6man-segment-routing-header] and [I-D.filsfils-spring-srv6-network-programming] also apply to this document.

This document does not introduce any new security vulnerabilities.

12. Acknowledgements

The authors would like to thank Thierry Couture, Ketan Talaulikar, Loa Andersson, Andrew G. Malis, Adrian Farrel, Alexander Vainshtein and Joel M. Halpern for their valuable comments and suggestions on the document.

13. Contributors

P. Camarillo (Cisco), B. Peirens (Proximus), D. Steinberg (Steinberg Consulting), A. AbdelSalam (Gran Sasso Science Institute), G. Dawra (LinkedIn), S. Bryant (Huawei), H. Assarpour (Broadcom), H. Shah (Ciena), L. Contreras (Telefonica I+D), J. Tantsura (Individual), M. Vigoureux (Nokia) and J. Bhattacharya (Cisco) substantially contributed to the content of this document.

14. References

14.1. Normative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Li, Z., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., and M. Sharif, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-04 (work in progress), March 2018.

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-14 (work in progress), June 2018.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[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-14 (work in progress), June 2018.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-01 (work in progress), June 2018.

14.2. Informative References

[I-D.dawra-idr-bgp-sr-service-chaining]
Dawra, G., Filsfils, C., daniel.bernier@bell.ca, d., Uttaro, J., Decraene, B., Elmalky, H., Xu, X., Clad, F., and K. Talaulikar, "BGP Control Plane Extensions for Segment Routing based Service Chaining", draft-dawra-idr-bgp-sr-service-chaining-02 (work in progress), January 2018.

[I-D.filsfils-spring-sr-policy-considerations]
Filsfils, C., Talaulikar, K., Krol, P., Horneffer, M., and P. Mattes, "SR Policy Implementation and Deployment Considerations", draft-filsfils-spring-sr-policy-considerations-01 (work in progress), June 2018.

[I-D.ietf-mpls-sfc]
Farrel, A., Bryant, S., and J. Drake, "An MPLS-Based Forwarding Plane for Service Function Chaining", draft-ietf-mpls-sfc-01 (work in progress), May 2018.

[I-D.xu-mpls-payload-protocol-identifier]
Xu, X., Assarpour, H., and S. Ma, "MPLS Payload Protocol Identifier", draft-xu-mpls-payload-protocol-identifier-04 (work in progress), January 2018.

[IFIP18] Abdelsalam, A., Salsano, S., Clad, F., Camarillo, P., and C. Filsfils, "SEgment Routing Aware Firewall For Service Function Chaining scenarios", IFIP Networking conference , May 2018.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

Authors' Addresses

Francois Clad (editor)
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Xiaohu Xu (editor)
Alibaba

Email: xiaohu.xxh@alibaba-inc.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Cheng Li
Huawei

Email: chengli13@huawei.com

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Shaowen Ma
Juniper

Email: mashaowen@gmail.com

Chaitanya Yadlapalli
AT&T
USA

Email: cy098d@att.com

Wim Henderickx
Nokia
Belgium

Email: wim.henderickx@nokia.com

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

SPRING
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

F. Clad, Ed.
Cisco Systems, Inc.
X. Xu, Ed.
Alibaba
C. Filsfils
Cisco Systems, Inc.
D. Bernier
Bell Canada
C. Li
Huawei
B. Decraene
Orange
S. Ma
Juniper
C. Yadlapalli
AT&T
W. Henderickx
Nokia
S. Salsano
Universita di Roma "Tor Vergata"

October 22, 2018

Service Programming with Segment Routing
draft-xuclad-spring-sr-service-programming-01

Abstract

This document defines data plane functionality required to implement service segments and achieve service programming in SR-enabled MPLS and IP networks, as described in the Segment Routing architecture.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 25, 2019.

Table of Contents

1. Introduction

Segment Routing (SR) is an architecture based on the source routing paradigm that seeks the right balance between distributed intelligence and centralized programmability. SR can be used with an MPLS or an IPv6 data plane to steer packets through an ordered list of instructions, called segments. These segments may encode simple routing instructions for forwarding packets along a specific network path, but also steer them through VNFs or physical service appliances available in the network.

In an SR network, each of these services, running either on a physical appliance or in a virtual environment, are associated with a segment identifier (SID). These service SIDs are then leveraged as part of a SID-list to steer packets through the corresponding services. Service SIDs may be combined together in a SID-list to achieve service programming, but also with other types of segments as defined in [RFC8402]. SR thus provides a fully integrated solution for overlay, underlay and service programming. Furthermore, the IPv6 instantiation of SR (SRv6) supports metadata transportation in the Segment Routing header [I-D.ietf-6man-segment-routing-header], either natively in the tag field or with extensions such as TLVs.

This document describes how a service can be associated with a SID, including legacy services with no SR capabilities, and how these service SIDs are integrated within an SR policy. The definition of an SR Policy and the traffic steering mechanisms are covered in [I-D.ietf-spring-segment-routing-policy] and hence outside the scope of this document.

The definition of control plane components, such as service segment discovery, is outside the scope of this data plane document. For reference, the option of using BGP extensions to support SR service programming is proposed in [I-D.dawra-idr-bgp-sr-service-chaining].

2. Terminology

This document leverages the terminology proposed in [RFC8402] and [I-D.ietf-spring-segment-routing-policy]. It also introduces the following new terms.

Service segment: A segment associated with a service. The service may either run on a physical appliance or in a virtual environment such as a virtual machine or container.

SR-aware service: A service that is fully capable of processing SR traffic. An SR-aware service can be directly associated with a service segment.

SR-unaware service: A service that is unable to process SR traffic or may behave incorrectly due to presence of SR information in the packet headers. An SR-unaware service can be associated with a service segment through an SR proxy function.

3. Classification and steering

Classification and steering mechanisms are defined in section 8 of [I-D.ietf-spring-segment-routing-policy] and are independent from the purpose of the SR policy. From the perspective of a headend node classifying and steering traffic into an SR policy, there is no difference whether this policy contains IGP, BGP, peering, VPN or service segments, or any combination of these.

As documented in the above reference, traffic is classified when entering an SR domain. The SR policy headend may, depending on its capabilities, classify the packets on a per-destination basis, via simple FIB entries, or apply more complex policy routing rules requiring to look deeper into the packet. These rules are expected to support basic policy routing such as 5-tuple matching. In addition, the IPv6 SRH tag field defined in [I-D.ietf-6man-segment-routing-header] can be used to identify and classify packets sharing the same set of properties. Classified traffic is then steered into the appropriate SR policy and forwarded as per the SID-list(s) of the active candidate path.

SR traffic can be re-classified by an SR endpoint along the original SR policy (e.g., DPI service) or a transit node intercepting the traffic. This node is the head-end of a new SR policy that is imposed onto the packet, either as a stack of MPLS labels or as an IPv6 SRH.

4. Service segments

In the context of this document, the term service refers to a physical appliance running on dedicated hardware, a virtualized service inside an isolated environment such as a VM, container or namespace, or any process running on a compute element. A service may also comprise multiple sub-components running in different processes or containers. Unless otherwise stated, this document does not make any assumption on the type or execution environment of a service.

The execution of a service can be integrated as part of an SR policy by assigning a segment identifier, or SID, to the service and including this service SID in the SR policy SID-list. Such a service SID may be of local or global significance. In the former case, other segments, such as prefix or adjacency segments, can be used to steer the traffic up to the node where the service segment is instantiated. In the latter case, the service is directly reachable from anywhere in the routing domain. This is realized with SR-MPLS by assigning a SID from the global label block ([I-D.ietf-spring-segment-routing-mpls]), or with SRv6 by advertising the SID locator in the routing protocol ([I-D.filsfils-spring-srv6-network-programming]). It is up to the network operator to define the scope and reachability of each service SID. This decision can be based on various considerations such as infrastructure dynamicity, available control plane or orchestration system capabilities.

This document categorizes services in two types, depending on whether they are able to behave properly in the presence of SR information or not. These are respectively named SR-aware and SR-unaware services.

4.1. SR-aware services

An SR-aware service can process the SR information in the packets it receives. This means being able to identify the active segment as a local instruction and move forward in the segment list, but also that the service's own behavior is not hindered due to the presence of SR information. For example, an SR-aware firewall filtering SRv6 traffic based on its final destination must retrieve that information from the last entry in the SRH rather than the Destination Address field of the IPv6 header.

An SR-aware service is associated with a locally instantiated service segment, which is used to steer traffic through it.

If the service is configured to intercept all the packets passing through the appliance, the underlying routing system only has to

implement a default SR endpoint behavior (SR-MPLS node segment or SRv6 End function), and the corresponding SID will be used to steer traffic through the service.

If the service requires the packets to be directed to a specific virtual interface, networking queue or process, a dedicated SR behavior may be required to steer the packets to the appropriate location. The definition of such service-specific functions is out of the scope of this document.

SR-aware services also enable advanced network programming functionalities such as conditional branching and jumping to arbitrary SIDs in the segment list. In addition, SRv6 provides several ways of passing and exchanging information between services (e.g., SID arguments, tag field and TLVs). An example scenario involving these features is described in [IFIP18], which discusses the implementation of an SR-aware Intrusion Detection System.

Examples of SR-aware services are provided in section Section 8.1.

4.2. SR-unaware services

Any service that does not meet the above criteria for SR-awareness is considered as SR-unaware.

An SR-unaware service is not able to process the SR information in the traffic that it receives. It may either drop the traffic or take erroneous decisions due to the unrecognized routing information. In order to include such services in an SR policy, it is thus required to remove the SR information as well as any other encapsulation header before the service receives the packet, or to alter it in such a way that the service can correctly process the packet.

In this document, we define the concept of an SR proxy as an entity, separate from the service, that performs these modifications and handle the SR processing on behalf of a service. The SR proxy can run as a separate process on the service appliance, on a virtual switch or router on the compute node or on a different host.

An SR-unaware service is associated with a service segment instantiated on the SR proxy, which is used to steer traffic through the service. Section 6 describes several SR proxy behaviors to handle the encapsulation headers and SR information under various circumstances.

5. SR service policies

An SR service policy is an SR policy, as defined in [I-D.ietf-spring-segment-routing-policy], that includes at least one service. This service is represented in the SID-list by its associated service SID. In case the policy should include several services, the service traversal order is indicated by the relative position of each service SID in the SID-list. Using the mechanisms described in [I-D.ietf-spring-segment-routing-policy], it is possible to load balance the traffic over several services, or instances of the same service, by associating with the SR service policy a weighted set of SID-lists, each containing a possible sequence of service SIDs to be traversed. Similarly, several candidate paths can be specified for the SR service policy, each with its own set of SID-lists, for resiliency purposes.

Furthermore, binding SIDs (BSIDs) can be leveraged in the context of service policies to reduce the number of SIDs imposed by the headend, provide opacity between domains and improve scalability, as described in [I-D.filsfils-spring-sr-policy-considerations]. For example, a network operator may want a policy in its core domain to include services that are running in one of its datacenters. One option is to define an SR policy at ingress edge of the core domain that explicitly includes all the SIDs needed to steer the traffic through the core and in the DC, but that may result in a long SID-list and requires to update the ingress edge configuration every time the DC part of the policy is modified. Alternatively, a separate policy can be defined at the ingress edge of the datacenter with only the SIDs that needs to be executed there and its BSID included in the core domain policy. That BSID remains stable when the DC policy is modified and can even be shared among several core domain policies that would require the same type of processing in the DC.

This section describes how services can be integrated within an SR-MPLS or SRv6 service policy.

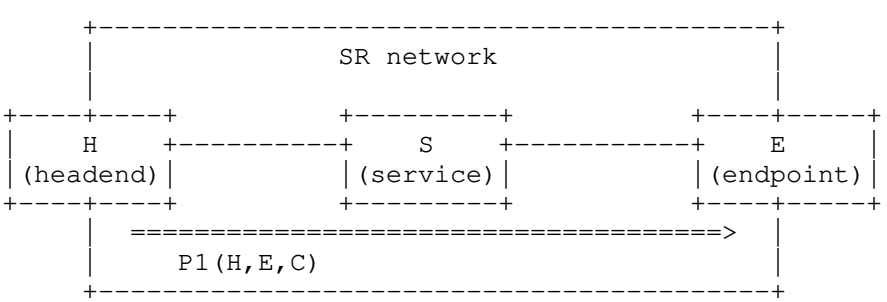


Figure 1: SR service policy

Figure 1 illustrates a basic SR service policy instantiated on a headend node H towards an endpoint E and traversing a service S. The SR policy may also include additional requirements, such as traffic engineering or VPN. On the head-end H, the SR policy P1 is created with a color C and endpoint E and associated with an SR path that can either be explicitly configured, dynamically computed on H or provisioned by a network controller.

In its most basic form, the SR policy P1 would be resolved into the SID-list < SID(S), SID(E) >. This is assuming that SID(S) and SID(E) are directly reachable from H and S, respectively, and that the forwarding path meets the policy requirement. However, depending on the dataplane and the segments available in the network, additional SIDs may be required to enforce the SR policy.

This model applies regardless of the SR-awareness of the service. If it is SR-unaware, then S simply represents the proxy that takes care of transmitting the packet to the actual service.

Traffic can then be steered into this policy using any of the mechanisms described in [I-D.ietf-spring-segment-routing-policy].

The following subsections describe the specificities of each SR dataplane.

5.1. SR-MPLS data plane

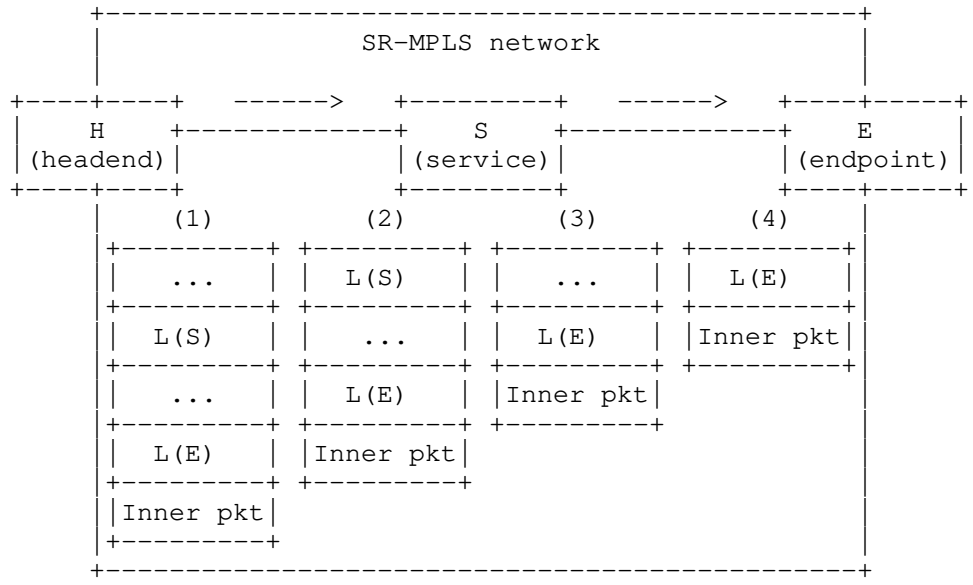


Figure 2: Packet walk in an SR-MPLS network

In an SR-MPLS network, the SR policy SID-list is encoded as a stack of MPLS labels[I-D.ietf-spring-segment-routing-mpls] and pushed on top of the packet.

In the example shown on Figure 2, the SR policy should steer the traffic from the head-end H to the endpoint E via a service S. This translates into an MPLS label stack that includes at least a label L(S) associated to service S and a label L(E) associated to the endpoint E. The label stack may also include additional intermediate segments if these are required for traffic engineering (e.g., to encode a low latency path between H and S and / or between S and E) or simply for reachability purposes. Indeed, the service SID L(S) may be taken from the global or local SID block of node S and, in the latter case, one or more SIDs might be needed before L(S) in order for the packet to reach node S (e.g., a prefix-SID of S), where L(S) can be interpreted. The same applies for the segment L(E) at the SR policy endpoint.

Special consideration must be taken into account when using Local SIDs for service identification due to increased label stack depth and the associated impacts.

When the packet arrives at S, this node determines how to process the packet based on the semantic locally associated to the top label L(S). If S is an SR-aware service, the SID L(S) may provide

additional context or indication on how to process the packet (e.g., payload type or a firewall SID may indicate which rule set should be applied onto the packet). If S is a proxy in front of an SR-unaware service, L(S) indicates how and to which service attached to this proxy the packet should be transmitted. At some point in the process, L(S) is also popped from the label stack in order to expose the next SID, which may be L(E) or another intermediate segment.

5.2. SRv6 data plane

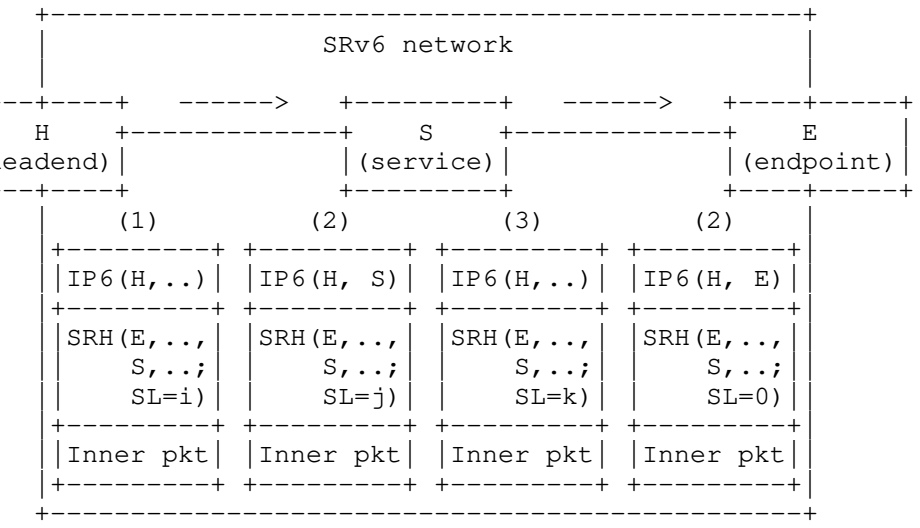


Figure 3: Packet walk in an SRv6 network

In an SRv6 network, the SR Policy is encoded into the packet as an IPv6 header possibly followed by a Segment Routing header (SRH) [I-D.ietf-6man-segment-routing-header].

In the example shown on Figure 3, the SR policy should steer the traffic from the head-end H to the endpoint E via a service S. This translates into an SRH that includes at least a segment SID(S) to the service, or service proxy, S and a segment SID(E) to the endpoint E. The SRH may also include additional intermediate segments if these are required for traffic engineering (e.g., the encode a low latency path between H and S and / or between S and E) or simply for reachability purposes. Indeed, the service segment locator may or may not be advertised in the routing protocol and, in the latter case, one or more SIDs might be needed before SID(S) in order to bring the packet up to node S, where SID(S) can be interpreted. The same applies for the segment SID(E) at the SR policy endpoint.

When the packet arrives at S, this node determines how to process the packet based on the semantic locally associated to the active segment SID(S). If S is an SR-aware service, then SID(S) may provide additional context or indication on how to process the packet (e.g., a firewall SID may indicate which rule set should be applied onto the packet). If S is a proxy in front of an SR-unaware service, SID(S) indicates how and to which service attached to this proxy the packet should be transmitted. At some point in the process, the SRv6 End function is also applied in order to make the next SID, which may be SID(E) or another intermediate segment, active.

The "Inner pkt" on Figure 3 represents the SRv6 payload, which may be an encapsulated IP packet, an Ethernet frame or a transport-layer payload, for example.

6. SR proxy behaviors

This section describes several SR proxy behaviors designed to enable SR service programming through SR-unaware services. A system implementing one of these functions may handle the SR processing on behalf of an SR-unaware service and allows the service to properly process the traffic that is steered through it.

A service may be located at any hop in an SR policy, including the last segment. However, the SR proxy behaviors defined in this section are dedicated to supporting SR-unaware services at intermediate hops in the segment list. In case an SR-unaware service is at the last segment, it is sufficient to ensure that the SR information is ignored (IPv6 routing extension header with Segments Left equal to 0) or removed before the packet reaches the service (MPLS PHP, SRv6 End.D or PSP).

As illustrated on Figure 4, the generic behavior of an SR proxy has two parts. The first part is in charge of passing traffic from the network to the service. It intercepts the SR traffic destined for the service via a locally instantiated service segment, modifies it in such a way that it appears as non-SR traffic to the service, then sends it out on a given interface, IFACE-OUT, connected to the service. The second part receives the traffic coming back from the service on IFACE-IN, restores the SR information and forwards it according to the next segment in the list. IFACE-OUT and IFACE-IN are respectively the proxy interface used for sending traffic to the service and the proxy interface that receives the traffic coming back from the service. These can be physical interfaces or sub-interfaces (VLANs) and, unless otherwise stated, IFACE-OUT and IFACE-IN can represent the same interface.

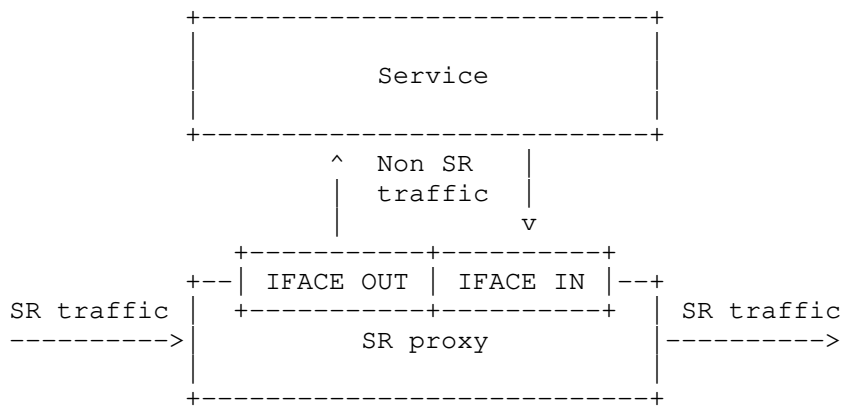


Figure 4: Generic SR proxy

In the next subsections, the following SR proxy mechanisms are defined:

- Static proxy
- Dynamic proxy
- Shared-memory proxy
- Masquerading proxy

Each mechanism has its own characteristics and constraints, which are summarized in the below table. It is up to the operator to select the best one based on the proxy node capabilities, the service behavior and the traffic type. It is also possible to use different proxy mechanisms within the same service policy.

		Static	Dynamic	Shared mem.	Masquerading
SR flavors	SR-MPLS	Y	Y	Y	-
	SRv6 insertion	P	P	P	Y
	SRv6 encapsulation	Y	Y	Y	-
Chain agnostic configuration		N	N	Y	Y
Transparent to chain changes		N	Y	Y	Y
Service support	DA modification	Y	Y	Y	NAT
	Payload modification	Y	Y	Y	Y
	Packet generation	Y	Y	cache	cache
	Packet deletion	Y	Y	Y	Y
	Transport endpoint	Y	Y	cache	cache
Supported traffic	Ethernet	Y	Y	Y	-
	IPv4	Y	Y	Y	-
	IPv6	Y	Y	Y	Y

Figure 5: SR proxy summary

Note: The use of a shared memory proxy requires both the service (VNF) and the proxy to be running on the same node.

6.1. Static SR proxy

The static proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware service. This proxy thus receives SR traffic that is formed of an MPLS label stack or an IPv6 header on top of an inner packet, which can be Ethernet, IPv4 or IPv6.

A static SR proxy segment is associated with the following mandatory parameters

- INNER-TYPE: Inner packet type
- NH-ADDR: Next hop Ethernet address (only for inner type IPv4 and IPv6)
- IFACE-OUT: Local interface for sending traffic towards the service
- IFACE-IN: Local interface receiving the traffic coming back from the service
- CACHE: SR information to be attached on the traffic coming back from the service, including at least
  - CACHE.SA: IPv6 source address (SRv6 only)
  - CACHE.LIST: Segment list expressed as MPLS labels or IPv6 address

A static SR proxy segment is thus defined for a specific service, inner packet type and cached SR information. It is also bound to a pair of directed interfaces on the proxy. These may be both directions of a single interface, or opposite directions of two different interfaces. The latter is recommended in case the service is to be used as part of a bi-directional SR SC policy. If the proxy and the service both support 802.1Q, IFACE-OUT and IFACE-IN can also represent sub-interfaces.

The first part of this behavior is triggered when the proxy node receives a packet whose active segment matches a segment associated with the static proxy behavior. It removes the SR information from the packet then sends it on a specific interface towards the associated service. This SR information corresponds to the full label stack for SR-MPLS or to the encapsulation IPv6 header with any attached extension header in the case of SRv6.

The second part is an inbound policy attached to the proxy interface receiving the traffic returning from the service, IFACE-IN. This

policy attaches to the incoming traffic the cached SR information associated with the SR proxy segment. If the proxy segment uses the SR-MPLS data plane, CACHE contains a stack of labels to be pushed on top of the packets. With the SRv6 data plane, CACHE is defined as a source address, an active segment and an optional SRH (tag, segments left, segment list and metadata). The proxy encapsulates the packets with an IPv6 header that has the source address, the active segment as destination address and the SRH as a routing extension header. After the SR information has been attached, the packets are forwarded according to the active segment, which is represented by the top MPLS label or the IPv6 Destination Address. An MPLS TTL or IPv6 Hop Limit value may also be configured in CACHE. If it is not, the proxy should set these values according to the node's default setting for MPLS or IPv6 encapsulation.

In this scenario, there are no restrictions on the operations that can be performed by the service on the stream of packets. It may operate at all protocol layers, terminate transport layer connections, generate new packets and initiate transport layer connections. This behavior may also be used to integrate an IPv4-only service into an SRv6 policy. However, a static SR proxy segment can be used in only one service policy at a time. As opposed to most other segment types, a static SR proxy segment is bound to a unique list of segments, which represents a directed SR SC policy. This is due to the cached SR information being defined in the segment configuration. This limitation only prevents multiple segment lists from using the same static SR proxy segment at the same time, but a single segment list can be shared by any number of traffic flows. Besides, since the returning traffic from the service is re-classified based on the incoming interface, an interface can be used as receiving interface (IFACE-IN) only for a single SR proxy segment at a time. In the case of a bi-directional SR SC policy, a different SR proxy segment and receiving interface are required for the return direction.

The static proxy behavior may also be used for sending traffic through "bump in the wire" services that are transparent to the IP and Ethernet layers. This type of processing is assumed when the inner traffic type is Ethernet, since the original destination address of the Ethernet frame is preserved when the packet is steered into the SR Policy and likely associated with a node downstream of the policy tail-end. In case the inner type is IP (IPv4 or IPv6), the NH-ADDR parameter may be set to a dummy or broadcast Ethernet address, or simply to the address of the proxy receiving interface (IFACE-IN).

6.1.1. SR-MPLS pseudocode

6.1.1.1. Static proxy for inner type Ethernet

Upon receiving an MPLS packet with top label L, where L is an MPLS L2 static proxy segment, a node N does:

```
1.  Pop all labels
2.  IF payload type is Ethernet THEN
3.      Forward the exposed frame on IFACE-OUT
4.  ELSE
5.      Drop the packet
```

Upon receiving on IFACE-IN an Ethernet frame with a destination address different than the interface address, a node N does:

```
1.  Push labels in CACHE on top of the frame Ethernet header
2.  Lookup the top label and proceed accordingly
```

The receiving interface must be configured in promiscuous mode in order to accept those Ethernet frames.

6.1.1.2. Static proxy for inner type IPv4

Upon receiving an MPLS packet with top label L, where L is an MPLS IPv4 static proxy segment, a node N does:

```
1.  Pop all labels
2.  IF payload type is IPv4 THEN
3.      Forward the exposed packet on IFACE-OUT towards NH-ADDR
4.  ELSE
5.      Drop the packet
```

Upon receiving a non-link-local IPv4 packet on IFACE-IN, a node N does:

```
1.  Decrement TTL and update checksum
2.  Push labels in CACHE on top of the packet IPv4 header
3.  Lookup the top label and proceed accordingly
```

6.1.1.3. Static proxy for inner type IPv6

Upon receiving an MPLS packet with top label L, where L is an MPLS IPv6 static proxy segment, a node N does:

```
1.   Pop all labels
2.   IF payload type is IPv6 THEN
3.       Forward the exposed packet on IFACE-OUT towards NH-ADDR
4.   ELSE
5.       Drop the packet
```

Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N does:

```
1.   Decrement Hop Limit
2.   Push labels in CACHE on top of the packet IPv6 header
3.   Lookup the top label and proceed accordingly
```

6.1.2. SRv6 pseudocode

6.1.2.1. Static proxy for inner type Ethernet

Upon receiving an IPv6 packet destined for S, where S is an IPv6 static proxy segment for Ethernet traffic, a node N does:

```
1.   IF ENH == 59 THEN                                          ;; Ref1
2.       Remove the (outer) IPv6 header and its extension headers
3.       Forward the exposed frame on IFACE-OUT
4.   ELSE
5.       Drop the packet
```

Ref1: 59 refers to "no next header" as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving on IFACE-IN an Ethernet frame with a destination address different than the interface address, a node N does:

```
1.   Retrieve CACHE entry matching IFACE-IN and traffic type
2.   Push SRH with CACHE.LIST on top of the Ethernet header   ;; Ref2
3.   Push IPv6 header with
       SA = CACHE.SA
       DA = CACHE.LIST[0]                                     ;; Ref3
       Next Header = 43                                       ;; Ref4
4.   Set outer payload length and flow label
5.   Lookup outer DA in appropriate table and proceed accordingly
```

Ref2: Unless otherwise specified, the segments in CACHE.LIST should be encoded in reversed order, Segment Left and Last Entry values should be set of the length of CACHE.LIST minus 1, and Next Header should be set to 59.

Ref3: CACHE.LIST[0] represents the first IPv6 SID in CACHE.LIST.

Ref4: If CACHE.LIST contains a single entry, the SRH can be omitted and the Next Header value must be set to 59.

The receiving interface must be configured in promiscuous mode in order to accept those Ethernet frames.

6.1.2.2. Static proxy for inner type IPv4

Upon receiving an IPv6 packet destined for S, where S is an IPv6 static proxy segment for IPv4 traffic, a node N does:

```
1.  IF ENH == 4 THEN                                          ;; Ref1
2.      Remove the (outer) IPv6 header and its extension headers
3.      Forward the exposed packet on IFACE-OUT towards NH-ADDR
4.  ELSE
5.      Drop the packet
```

Ref1: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving a non-link-local IPv4 packet on IFACE-IN, a node N does:

```
1.  Decrement TTL and update checksum
2.  IF CACHE.SRH THEN                                         ;; Ref2
3.      Push CACHE.SRH on top of the existing IPv4 header
4.      Set NH value of the pushed SRH to 4
5.  Push outer IPv6 header with SA, DA and traffic class from CACHE
6.  Set outer payload length and flow label
7.  Set NH value to 43 if an SRH was added, or 4 otherwise
8.  Lookup outer DA in appropriate table and proceed accordingly
```

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

6.1.2.3. Static proxy for inner type IPv6

Upon receiving an IPv6 packet destined for S, where S is an IPv6 static proxy segment for IPv6 traffic, a node N does:

```
1.  IF ENH == 41 THEN                                         ;; Ref1
2.      Remove the (outer) IPv6 header and its extension headers
3.      Forward the exposed packet on IFACE-OUT towards NH-ADDR
4.  ELSE
5.      Drop the packet
```

Ref1: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N does:

```
1.  Decrement Hop Limit
2.  IF CACHE.SRH THEN                                          ;; Ref2
3.      Push CACHE.SRH on top of the existing IPv6 header
4.      Set NH value of the pushed SRH to 41
5.  Push outer IPv6 header with SA, DA and traffic class from CACHE
6.  Set outer payload length and flow label
7.  Set NH value to 43 if an SRH was added, or 41 otherwise
8.  Lookup outer DA in appropriate table and proceed accordingly
```

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

6.2. Dynamic SR proxy

The dynamic proxy is an improvement over the static proxy that dynamically learns the SR information before removing it from the incoming traffic. The same information can then be re-attached to the traffic returning from the service. As opposed to the static SR proxy, no CACHE information needs to be configured. Instead, the dynamic SR proxy relies on a local caching mechanism on the node instantiating this segment.

Upon receiving a packet whose active segment matches a dynamic SR proxy function, the proxy node pops the top MPLS label or applies the SRv6 End behavior, then compares the updated SR information with the cache entry for the current segment. If the cache is empty or different, it is updated with the new SR information. The SR information is then removed and the inner packet is sent towards the service.

The cache entry is not mapped to any particular packet, but instead to an SR SC policy identified by the receiving interface (IFACE-IN). Any non-link-local IP packet or non-local Ethernet frame received on that interface will be re-encapsulated with the cached headers as described in Section 6.1. The service may thus drop, modify or generate new packets without affecting the proxy.

6.2.1. SR-MPLS pseudocode

The dynamic proxy SR-MPLS pseudocode is obtained by inserting the following instructions at the beginning of the static SR-MPLS pseudocode (Section 6.1.1).

```
1.   IF top label S bit is 0 THEN                                ;; Ref1
2.       Pop top label
3.       IF C(IFACE-IN) different from remaining labels THEN     ;; Ref2
4.           Copy all remaining labels into C(IFACE-IN)          ;; Ref3
5.   ELSE
6.       Drop the packet
```

Ref1: As mentioned at the beginning of Section 6, an SR proxy is not needed to include an SR-unaware service at the end of an SR policy.

Ref2: A TTL margin can be configured for the top label stack entry to prevent constant cache updates when multiple equal-cost paths with different hop counts are used towards the SR proxy node. In that case, a TTL difference smaller than the configured margin should not trigger a cache update (provided that the labels are the same).

Ref3: C(IFACE-IN) represents the cache entry associated to the dynamic SR proxy segment. It is identified with IFACE-IN in order to efficiently retrieve the right SR information when a packet arrives on this interface.

In addition, the inbound policy should check that C(IFACE-IN) has been defined before attempting to restore the MPLS label stack and drop the packet otherwise.

6.2.2. SRv6 pseudocode

The dynamic proxy SRv6 pseudocode is obtained by inserting the following instructions between lines 1 and 2 of the static proxy SRv6 pseudocode.

```
1.   IF NH=SRH & SL > 0 THEN                                    ;; Ref1
2.       Decrement SL and update the IPv6 DA with SRH[SL]
3.       IF C(IFACE-IN) different from IPv6 encaps THEN         ;; Ref2
4.           Copy the IPv6 encaps into C(IFACE-IN)              ;; Ref3
5.   ELSE
6.       Drop the packet
```

Ref1: As mentioned at the beginning of Section 6, an SR proxy is not needed to include an SR-unaware service at the end of an SR policy.

Ref2: "IPv6 encaps" represents the IPv6 header and any attached extension header.

Ref3: C(IFACE-IN) represents the cache entry associated to the dynamic SR proxy segment. It is identified with IFACE-IN in order to efficiently retrieve the right SR information when a packet arrives on this interface.

In addition, the inbound policy should check that C(IFACE-IN) has been defined before attempting to restore the IPv6 encapsulation and drop the packet otherwise.

6.3. Shared memory SR proxy

The shared memory proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware service. This proxy behavior leverages a shared-memory interface with a virtualized service (VNF) in order to hide the SR information from an SR-unaware service while keeping it attached to the packet. We assume in this case that the proxy and the VNF are running on the same compute node. A typical scenario is an SR-capable vrouter running on a container host and forwarding traffic to VNFs isolated within their respective container.

More details will be added in a future revision of this document.

6.4. Masquerading SR proxy

The masquerading proxy is an SR endpoint behavior for processing SRv6 traffic on behalf of an SR-unaware service. This proxy thus receives SR traffic that is formed of an IPv6 header and an SRH on top of an inner payload. The masquerading behavior is independent from the inner payload type. Hence, the inner payload can be of any type but it is usually expected to be a transport layer packet, such as TCP or UDP.

A masquerading SR proxy segment is associated with the following mandatory parameters:

- S-ADDR: Ethernet or IPv6 address of the service
- IFACE-OUT: Local interface for sending traffic towards the service
- IFACE-IN: Local interface receiving the traffic coming back from the service

A masquerading SR proxy segment is thus defined for a specific service and bound to a pair of directed interfaces or sub-interfaces on the proxy. As opposed to the static and dynamic SR proxies, a masquerading segment can be present at the same time in any number of SR SC policies and the same interfaces can be bound to multiple masquerading proxy segments. The only restriction is that a masquerading proxy segment cannot be the last segment in an SR SC policy.

The first part of the masquerading behavior is triggered when the proxy node receives an IPv6 packet whose Destination Address matches a masquerading proxy segment. The proxy inspects the IPv6 extension headers and substitutes the Destination Address with the last segment in the SRH attached to the IPv6 header, which represents the final destination of the IPv6 packet. The packet is then sent out towards the service.

The service receives an IPv6 packet whose source and destination addresses are respectively the original source and final destination. It does not attempt to inspect the SRH, as RFC8200 specifies that routing extension headers are not examined or processed by transit nodes. Instead, the service simply forwards the packet based on its current Destination Address. In this scenario, we assume that the service can only inspect, drop or perform limited changes to the packets. For example, Intrusion Detection Systems, Deep Packet Inspectors and non-NAT Firewalls are among the services that can be supported by a masquerading SR proxy. Variants of the masquerading behavior are defined in Section 6.4.2 and Section 6.4.3 to support a wider range of services.

The second part of the masquerading behavior, also called de-masquerading, is an inbound policy attached to the proxy interface receiving the traffic returning from the service, IFACE-IN. This policy inspects the incoming traffic and triggers a regular SRv6 endpoint processing (End) on any IPv6 packet that contains an SRH. This processing occurs before any lookup on the packet Destination Address is performed and it is sufficient to restore the right active segment as the Destination Address of the IPv6 packet.

6.4.1. SRv6 masquerading proxy pseudocode

Masquerading: Upon receiving a packet destined for S, where S is an IPv6 masquerading proxy segment, a node N processes it as follows.

```
1.   IF NH=SRH & SL > 0 THEN
2.       Update the IPv6 DA with SRH[0]
3.       Forward the packet on IFACE-OUT
4.   ELSE
5.       Drop the packet
```

De-masquerading: Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N processes it as follows.

```
1.   IF NH=SRH & SL > 0 THEN
2.       Decrement SL
3.       Update the IPv6 DA with SRH[SL]                    ;; Ref1
4.       Lookup DA in appropriate table and proceed accordingly
```

Ref1: This pseudocode can be augmented to support the Penultimate Segment Popping (PSP) endpoint flavor. The exact pseudocode modification are provided in [I-D.filsfils-spring-srv6-network-programming].

6.4.2. Variant 1: Destination NAT

Services modifying the destination address in the packets they process, such as NATs, can be supported by a masquerading proxy with the following modification to the de-masquerading pseudocode.

De-masquerading - NAT: Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N processes it as follows.

```
1.   IF NH=SRH & SL > 0 THEN
2.       Update SRH[0] with the IPv6 DA
3.       Decrement SL
4.       Update the IPv6 DA with SRH[SL]
5.       Lookup DA in appropriate table and proceed accordingly
```

6.4.3. Variant 2: Caching

Services generating packets or acting as endpoints for transport connections can be supported by adding a dynamic caching mechanism similar to the one described in Section 6.2.

More details will be added in a future revision of this document.

7. Metadata

7.1. MPLS data plane

Metadata can be carried for SR-MPLS traffic in a Segment Routing header inserted between the last MPLS label and the MPLS payload. When used solely as a metadata container, the SRH does not carry any segment but only the mandatory header fields, including the tag and flags, and any TLVs that is required for transporting the metadata.

Since the MPLS encapsulation has no explicit protocol identifier field to indicate the protocol type of the MPLS payload, how to indicate the presence of metadata in an MPLS packet is a potential issue to be addressed. One possible solution is to add the indication about the presence of metadata in the semantic of the SIDs. Note that only the SIDs whose behavior involves looking at the metadata or the MPLS payload would need to include such semantic (e.g., service segments). Other segments, such as traffic engineering segments, are not affected by the presence of metadata. Another, more generic, solution is to introduce a protocol identifier

field within the MPLS packet as described in [I-D.xu-mpls-payload-protocol-identifier].

7.2. IPv6 data plane

7.2.1. SRH TLV objects

The IPv6 SRH TLV objects are designed to carry all sorts of metadata. In particular, the NSH carrier TLV is defined as a container for NSH metadata.

TLV objects can be imposed by the ingress edge router that steers the traffic into the SR SC policy.

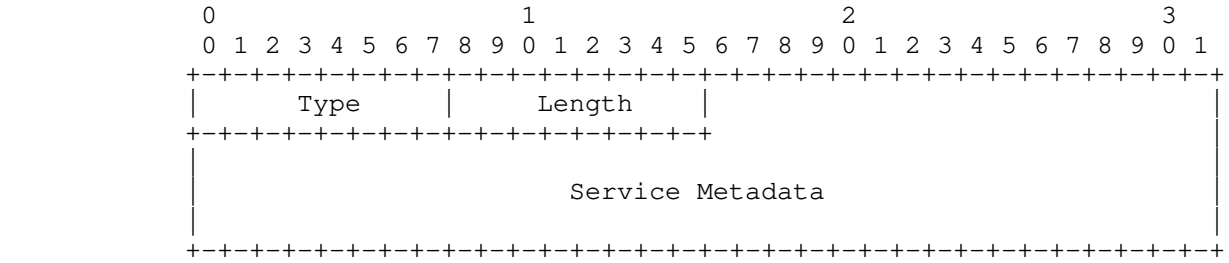
An SR-aware service may impose, modify or remove any TLV object attached to the first SRH, either by directly modifying the packet headers or via a control channel between the service and its forwarding plane.

An SR-aware service that re-classifies the traffic and steers it into a new SR SC policy (e.g. DPI) may attach any TLV object to the new SRH.

Metadata imposition and handling will be further discussed in a future version of this document.

7.2.1.1. Opaque Metadata TLV

This document defines an SRv6 TLV called Opaque Metadata TLV. This is a fixed-length container to carry any type of Service Metadata. No assumption is made by this document on the structure or the content of the carried metadata. The Opaque Metadata TLV has the following format:



where:

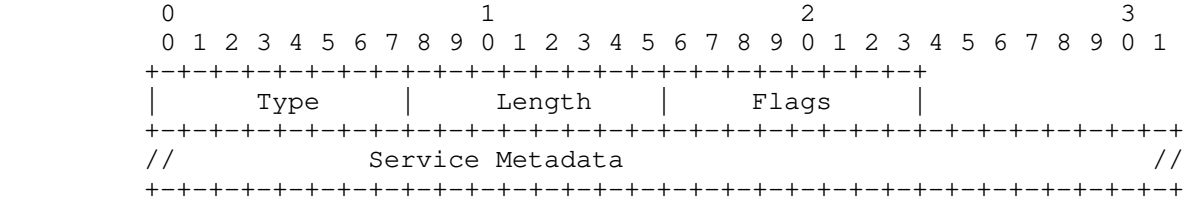
- Type: to be assigned by IANA.

- Length: 14.

- Service Metadata: 14 octets of opaque data.

7.2.1.2. NSH Carrier TLV

This document defines an SRv6 TLV called NSH Carrier TLV. It is a container to carry Service Metadata in the form of Variable-Length Metadata as defined in [RFC8300] for NSH MD Type 2. The NSH Carrier TLV has the following format:



where:

- Type: to be assigned by IANA.

- Length: the total length of the TLV.

- Flags: 8 bits. No flags are defined in this document. SHOULD be set to 0 on transmission and MUST be ignored on receipt.

- Service Metadata: a list of Service Metadata TLV as defined in [RFC8300] for NSH MD Type 2.

7.2.2. SRH tag

The SRH tag identifies a packet as part of a group or class of packets [I-D.ietf-6man-segment-routing-header].

In the context of service programming, this field can be used to encode basic metadata in the SRH. An example use case would be to leverage the SRH tag to encode a policy ID which could be leveraged in an SR-aware function to determine which processing policy to apply rather than having doing local classification or leverage alternate encapsulations.

8. Implementation status

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

8.1. SR-aware services

Specific SRv6 support has been implemented for the below open-source services:

- Iptables (1.6.2 and later)
- Nftables (0.8.4 and later)
- Snort

In addition, any service relying on the Linux kernel, version 4.10 and later, or FD.io VPP for packet forwarding can be considered as SR-aware.

8.2. Proxy behaviors

The static SR proxy is available for SR-MPLS and SRv6 on various Cisco hardware and software platforms. Furthermore, the following proxies are available on open-source software.

		VPP	Linux
MPLS	Static proxy	Available	In progress
	Dynamic proxy	In progress	In progress
	Shared memory proxy	In progress	In progress
SRv6	Static proxy	Available	In progress
	Dynamic proxy	Available	Available
	Shared memory proxy	In progress	In progress
	Masquerading proxy	Available	Available

Figure 6: Open-source implementation status table

9. Related works

The Segment Routing solution addresses a wide problem that covers both topological and service policies. The topological and service instructions can be either deployed in isolation or in combination. SR has thus a wider applicability than the architecture defined in [RFC7665]. Furthermore, the inherent property of SR is a stateless

network fabric. In SR, there is no state within the fabric to recognize a flow and associate it with a policy. State is only present at the ingress edge of the SR domain, where the policy is encoded into the packets. This is completely different from other proposals such as [RFC8300] and the MPLS label swapping mechanism described in [I-D.ietf-mpls-sfc], which rely on state configured at every hop of the service chain.

10. IANA Considerations

10.1. SRv6 Endpoint Behaviors

This I-D requests the IANA to allocate, within the "SRv6 Endpoint Behaviors" sub-registry belonging to the top-level "Segment-routing with IPv6 dataplane (SRv6) Parameters" registry, the following allocations:

Value	Description	Reference
TBA1	End.AN - SR-aware function (native)	[This.ID]
TBA2	End.AS - Static proxy	[This.ID]
TBA3	End.AD - Dynamic proxy	[This.ID]
TBA4	End.AM - Masquerading proxy	[This.ID]

10.2. Segment Routing Header TLVs

This I-D requests the IANA to allocate, within the "Segment Routing Header TLVs" registry, the following allocations:

Value	Description	Reference
TBA1	Opaque Metadata TLV	[This.ID]
TBA2	NSH Carrier TLV	[This.ID]

11. Security Considerations

The security requirements and mechanisms described in [RFC8402], [I-D.ietf-6man-segment-routing-header] and [I-D.filsfils-spring-srv6-network-programming] also apply to this document.

This document does not introduce any new security vulnerabilities.

12. Acknowledgements

The authors would like to thank Thierry Couture, Ketan Talaulikar, Loa Andersson, Andrew G. Malis, Adrian Farrel, Alexander Vainshtein

and Joel M. Halpern for their valuable comments and suggestions on the document.

13. Contributors

P. Camarillo (Cisco), B. Peirens (Proximus), D. Steinberg (Steinberg Consulting), A. AbdelSalam (Gran Sasso Science Institute), G. Dawra (LinkedIn), S. Bryant (Huawei), H. Assarpour (Broadcom), H. Shah (Ciena), L. Contreras (Telefonica I+D), J. Tantsura (Individual), M. Vigoureux (Nokia) and J. Bhattacharya (Cisco) substantially contributed to the content of this document.

14. References

14.1. Normative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-05 (work in progress), July 2018.

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-14 (work in progress), June 2018.

[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-14 (work in progress), June 2018.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-01 (work in progress), June 2018.

[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>.

14.2. Informative References

[I-D.dawra-idr-bgp-sr-service-chaining]
Dawra, G., Filsfils, C., daniel.bernier@bell.ca, d., Uttaro, J., Decraene, B., Elmalky, H., Xu, X., Clad, F., and K. Talaulikar, "BGP Control Plane Extensions for Segment Routing based Service Chaining", draft-dawra-idr-bgp-sr-service-chaining-02 (work in progress), January 2018.

[I-D.filsfils-spring-sr-policy-considerations]
Filsfils, C., Talaulikar, K., Krol, P., Horneffer, M., and P. Mattes, "SR Policy Implementation and Deployment Considerations", draft-filsfils-spring-sr-policy-considerations-02 (work in progress), October 2018.

[I-D.ietf-mpls-sfc]
Farrel, A., Bryant, S., and J. Drake, "An MPLS-Based Forwarding Plane for Service Function Chaining", draft-ietf-mpls-sfc-03 (work in progress), October 2018.

[I-D.xu-mpls-payload-protocol-identifier]
Xu, X., Assarpour, H., Ma, S., and F. Clad, "MPLS Payload Protocol Identifier", draft-xu-mpls-payload-protocol-identifier-05 (work in progress), August 2018.

[IFIP18] Abdelsalam, A., Salsano, S., Clad, F., Camarillo, P., and C. Filsfils, "SEgment Routing Aware Firewall For Service Function Chaining scenarios", IFIP Networking conference , May 2018.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

Authors' Addresses

Francois Clad (editor)
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Xiaohu Xu (editor)
Alibaba

Email: xiaohu.xxh@alibaba-inc.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Cheng Li
Huawei

Email: chengli13@huawei.com

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Shaowen Ma
Juniper

Email: mashaowen@gmail.com

Chaitanya Yadlapalli
AT&T
USA

Email: cy098d@att.com

Wim Henderickx
Nokia
Belgium

Email: wim.henderickx@nokia.com

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it