Bidirectional Forwarding Detection (BFD) for Segment Routing Policies for Traffic Engineering
draft-ali-spring-bfd-sr-policy-00

Abstract

Segment Routing (SR) allows a headend node to steer a packet flow along any path using a segment list which is referred to as a SR Policy. Intermediate per-flow states are eliminated thanks to source routing. The header of a packet steered in an SR Policy is augmented with the ordered list of segments associated with that SR Policy. Bidirectional Forwarding Detection (BFD) is used to monitor different kinds of paths between node. BFD mechanisms can be also used to monitor the availability of the path indicated by a SR Policy and to detect any failures. Seamless BFD (SBFD) extensions provide a simplified mechanism which is suitable for monitoring of paths that are setup dynamically and on a large scale.

This document describes the use of Seamless BFD (SBFD) mechanism to monitor the SR Policies that are used for Traffic Engineering (TE) in SR deployments.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction ......................................................... 2
2. Choice of SBFD over BFD ........................................... 3
3. Procedures ............................................................... 4
4. IANA Considerations ................................................ 5
5. Security Considerations ............................................. 6
6. Contributors ............................................................... 6
7. Acknowledgements .................................................... 6
8. References ............................................................... 6
   8.1. Normative References ........................................... 6
   8.2. Informative References ......................................... 7
Authors’ Addresses ............................................................. 8

1. Introduction

Segment Routing (SR) ([I-D.ietf-spring-segment-routing]) allows a headend node to steer a packet flow along any path for specific objectives like Traffic Engineering (TE) and to provide it treatment according to the specific established service level agreement (SLA) for it. 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 the ordered list of segments associated with that SR Policy. SR Policy [I-D.filsfils-spring-segment-routing-policy] specifies the concepts of SR Policy and steering into an SR Policy.
SR Policy state is instantiated only on the head-end node and any intermediate node or the endpoint node does not require any state to be maintained or instantiated for it. SR Policies are not signaled through the network nodes except the signaling required to instantiate them on the head-end in the case of a controller based deployment. This enables SR Policies to scale far better than previous TE mechanisms. This also enables SR Policies to be instantiated dynamically and on demand basis for steering specific traffic flows corresponding to service routes as they are signaled. These automatic steering and signaling mechanisms for SR Policies are described in SR Policy [I-D.filsfils-spring-segment-routing-policy].

There is a requirement to continuously monitor the availability of the path corresponding to the SR Policy along the nodes in the network and to signal any failures detected to the head-end node so that it could take corrective action to restore service. The corrective actions may be either to invalidate the candidate path that has experienced failure and to switch to another candidate path within the same SR Policy OR to activate another backup SR Policy or candidate path for end-to-end path protection. These mechanisms are beyond the scope of this document.

Bidirectional Forwarding Detection (BFD) mechanisms have been specified for use for monitoring of unidirectional MPLS LSPs via BFD MPLS [RFC5884]. Seamless BFD [RFC7880] defines a simplified mechanism for using BFD with a large proportion of negotiation aspects eliminated, thus providing benefits such as quick provisioning, as well as improved control and flexibility for network nodes initiating path monitoring. When BFD or SBFD is used for verification of such unidirectional LSP paths, the reverse path is via the shortest path from the tail-end router back to the head-end router as determined by routing.

The SR Policy is essentially a unidirectional path through the network. This document describes the use of BFD and more specifically SBFD for monitoring of SR Policy paths through the network. SR can be instantiated using both MPLS and IPv6 dataplanes. The mechanism described in this document applies to both these instantiations of SR Policy.

2. Choice of SBFD over BFD

BFD MPLS [RFC5884] describes a mechanism where LSP Ping [RFC8029] is used to bootstrap the BFD session over an MPLS TE LSP path. The LSP Ping mechanism was extended to support SR LSPs via SR LSP Ping [RFC8287] and a similar mechanism could have been considered for BFD monitoring of SR Policies on MPLS data-plane. However, this document
proposes instead to use SBFD mechanism as it is more suitable for SR Policies.

Some of the key aspects of SR Policies that are considered in arriving at this decision are as follows:

- SR Policies do not require any signaling to be performed through the network nodes in order to be setup. They are simply instantiated on the head-end node via provisioning or even dynamically by a controller via BGP SR-TE [I-D.ietf-idr-segment-routing-te-policy] or using PCEP (PCEP SR [I-D.ietf-pce-segment-routing], PCE Initiated [RFC8281], PCEP Stateful [RFC8231]).

- SR Policies result in state being instantiated only on the head-end node and no other node in the network.

- In many deployments, SR Policies are instantiated dynamically and on-demand or in the case of automated steering for BGP routes, when routes are learnt with specific color communities (refer SR Policy [I-D.filsfils-spring-segment-routing-policy] for details).

- SR Policies are expected to be deployed in much higher scale.

- SR Policies can be instantiated both for MPLS and IPv6 data-planes and hence a monitoring mechanism which works for both is desirable.

In view of the above, the BFD mechanism to be used for monitoring them needs to be simple, lightweight, one that does not result in instantiation of per SR Policy state anywhere but the head-end and which can be setup and deleted dynamically, on-demand and at scale. The SFBD extensions provide this support as described in Seamless BFD [RFC7880]. Furthermore, SBFD Use-Cases [RFC7882] clarifies the applicability in the Centralized TE and SR scenarios.

3. Procedures

The general procedures and mechanisms for SBFD operations are specified in Seamless BFD [RFC7880]. This section describes the specifics related to SBFD use for SR Policies.

SR Policies are represented on a head-end router as <color,endpoint IP address> tuple. The SRTE process on the head-end determines the tail-end node of a SR Policy on the basis of the endpoint IP address. In the cases where the SR Policy endpoint is outside the domain of the head-end node, this information is available with the centralized
controller that computed the multi-domain SR Policy path for the head-end.

In order to enable SBFD monitoring for a given SR Policy, the SBFD Discriminator for the tail-end node (i.e. one with the endpoint IP address) which is going to be the SBFD Reflector is required. ISIS SBFD [RFC7883] and OSPF SBFD [RFC7884] describe the extensions to the ISIS and OSPF link state routing protocols that allow all nodes to advertise their SBFD Discriminators across the network. BGP-LS SBFD [I-D.li-idr-bgp-ls-sbfd-extensions] describes extensions for advertising the SBFD discriminators via BGP-LS across domains and to a controller. Thus, either the SRTE head-end node or the controller, as the case may be, have the SBFD Discriminator of the tail-end node of the SR Policy available.

The SRTE Process can straightaway instantiate the SBFD mechanism on the SR Policy as soon as it is provisioned in the forwarding to start verification of the path to the endpoint. No signaling or provisioning is required for the tail-end node on a per SR Policy basis and it just performs its role as a stateless SBFD Reflector. The return path used by SBFD is via the normal IP routing back to the head-end node. Once the specific SR Policy path is verified via SBFD, then it is considered as active and may be used for traffic steering.

The SBFD monitoring continues for the SR Policy and any failure is notified to the SRTE process. In response to the failure of a specific candidate path, the SRTE process may trigger any of the following based on local policy or implementation specific aspects which are outside the scope of this document:

- Trigger path-protection for the SR Policy
- Declare the specific candidate path as invalid and switch to using the next valid candidate path based on preference
- If no alternate candidate path is available, then handle the steering over that SR Policy based on its invalidation policy (e.g. drop or switch to best effort routing).

4. IANA Considerations

None
5. Security Considerations

Procedures described in this document do not affect the BFD or Segment Routing security model. See the 'Security Considerations' section of [RFC7880] for a discussion of SBFD security and to [I-D.ietf-spring-segment-routing] for analysis of security in SR deployments.

6. Contributors

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

Mallik Mudigonda
Cisco Systems Inc.
Email: mmudigon@cisco.com

7. Acknowledgements

8. References

8.1. Normative References

[I-D.filsfils-spring-segment-routing-policy]

[I-D.ietf-spring-segment-routing]

[I-D.li-idr-bgp-ls-sbfd-extensions]
8.2. Informative References

[I-D.ietf-idr-segment-routing-te-policy]

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

[RFC5884]


Authors’ Addresses

Zafar Ali
Cisco Systems

Email: zali@cisco.com

Ketan Talaulikar
Cisco Systems

Email: ketant@cisco.com

Clarence Filsfils
Cisco Systems

Email: cfilsfil@cisco.com
Abstract

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

Status of This Memo

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

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

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

Copyright Notice

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

document authors. All rights reserved.

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

Table of Contents

1. Introduction .................................................. 3
2. Conventions Used in This Document ............................ 3
   2.1. Abbreviations ............................................. 3
   2.2. Terminology and Reference Topology ....................... 3
3. OAM Mechanisms ................................................. 4
   3.1. Ping ...................................................... 5
      3.1.1. Classic Ping ........................................... 5
      3.1.2. Pinging SID Function ................................. 6
            3.1.2.1. End-to-end Ping Using END.OTP ............... 7
      3.1.2.2. Segment-by-segment Ping Using O-bit (Proof of Transit) ........................................... 8
   3.2. Error Reporting ........................................... 9
3.3. Traceroute ................................................... 10
   3.3.1. Classic Traceroute ..................................... 10
   3.3.2. Traceroute to a SID Function .......................... 11
      3.3.2.1. Hop-by-hop Traceroute Using END.OTP .......... 12
      3.3.2.2. Tracing SRv6 Overlay .............................. 14
4. In-situ OAM Applicability ........................................ 15
5. Seamless BFD Applicability ....................................... 16
6. Monitoring of SRv6 Paths ......................................... 16
7. Security Considerations ......................................... 17
8. IANA Considerations ............................................. 17
9. References ...................................................... 17
   9.1. Normative References ..................................... 17
   9.2. Informative References ................................... 18
10. Acknowledgments ................................................ 20
Authors’ Addresses .................................................. 20
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.

```
+------------------------| N100 |------------------------+
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
====== link1====== link3------ link5====== link9------||N1||======||N2||======| N3 |======||N4||======| N5 |
|                      |      |------|  ||------|    |------|  ||------|    |
====== link2====== link4------ link6======link10------|
|                      |        ------        |
+--------| N6 |--------+                          |
        link7 |    | link8                        |
```

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::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::Ci represents 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.filisfils-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 responds 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 0-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
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.

- 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.

- 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:

- 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).

- When node N2 receives the packet with hop-count = 1, it processes
the hop count expiry. Specifically, the node N2 responds 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 responds 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 punt 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 punt 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.

```plaintext
> 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
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


9.2. Informative References


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
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.
document authors. All rights reserved.

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

Table of Contents

1. Introduction......................................................3
2. Conventions Used in This Document................................3
   2.1. Abbreviations.............................................3
   2.2. Terminology and Reference Topology........................4
3. OAM Building Blocks...............................................5
   3.1. O-flag in Segment Routing Header..........................5
   3.1.1. O-flag Processing....................................6
   3.1.2. Disabling Penultimate Segment Pop (PSP)..............7
   3.2. OAM Segments..............................................7
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.
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.

```
+--------------------------| N100 |------------------------+
|                           |
====== link1====== link3------ link5====== link9------
|     |     |     |     |     |     |     |     |
|N1|     |N2|     |N3|     |N4|     |N5|
====== link2====== link4------ link6====== link10------
|                           |
+-------| N6 |---------+
link7 |    | link8
------
```

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>.

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 SRH contains an 8-bit "Flags" field [I-D. draft-ietf-6man-segment-routing-header]. This document defines the following bit in the SRH.Flags to carry the O-flag:

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

3.1.1. O-flag Processing

Implementation of the O-flag is OPTIONAL. A node MAY ignore SRH.Flags.O-flag. It is also possible that a node is capable of supporting the O-bit but based on a local decision it MAY ignore it during processing on some local SIDs. If a node does not support the O-flag, then upon reception it simply ignores it. If a node supports the O-flag, it can optionally advertise its potential via node capability advertisement in IGP [I-D.bashandy-isis-srv6-extensions] and BGP-LS [I-D.dawra-idr-bgpls-srv6-ext].

The SRH.Flags.O-flag implements the "punt a timestamped copy and forward" behavior. To avoid the head of the line processing of the packet, some implementation may implement the "forward and punt a timestamped copy" behavior, instead. In order to implement "punt a timestamped copy and forward" or "forward and punt a timestamped copy" behavior, the following instructions are inserted at the beginning or the end of the pseudo-code for all SID Functions, respectively.

When N receives a packet whose IPv6 DA is S and S is a local SID, N executes the following the pseudo-code, either before or after the execution of the local SID S.

1. IF SRH.Flags.O-flag is True and SRH.Flags.O-flag is locally supported for S THEN
   a. Timestamp a local copy of the packet. ;; Ref1
   b. Punt the time-stamped copy of the packet to CPU for processing in software (slow-path). ;; Ref2

Ref1: Timestamping is done in hardware, as soon as possible during the packet processing. As timestamping is done on a copy of the packet which is locally punted, timestamp value can be carried in the local packet (punt) header.
Ref1: Hardware (microcode) just punts the packet. There is no requirement for the hardware to manipulate any TLV in SRH (or
3.1.2. Disabling Penultimate Segment Pop (PSP)

Penultimate Segment Pop (PSP) needs to be disabled when SRH.Flags.O-flag is set. If a node supports SRH.Flags.O-flag, it adds the following check after executing the instruction ‘update the IPv6 DA with SRH[SL]’ during processing of a local SID as described in [I-D.draft-filsfils-spring-srv6-network-programming]:

1. IF updated SL = 0 & PSP is TRUE and SRH.Flags.O-bit is False
2. pop the top SRH              ;; Ref1

Ref1: PSP behavior is disabled when SRH.Flags.O-flag is set.

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. As timestamping is done on a copy of the packet which is locally punked, timestamp value can be carried in the local packet (punt) header.

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 <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 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 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 (see [I-D.filsfils-spring-srv6-network-programming] for details). 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 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 END.OTP SID 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 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 A4::C52) is locally programmed.
- If the target SID is not locally programmed, N4 responds with the ICMPv6 message (Type: "SRv6 OAM (TBA)", Code: "SID not locally implemented (TBA)"); otherwise a success is returned.

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

Consider the same example where the user wants to ping a remote SID function 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-flag in 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, 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-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 A2::C31 SID function on the original packet and forwards it, accordingly. As SRH.Flags.O=1, Node N2 also disables the PSP flavour, 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 (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, A2::C31.
- 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.
- 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-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 (A2::C31) is locally programmed or not and responds to the ICMPv6 Echo Request. If the target SID is not locally programmed, N4 responds with the ICMPv6 message (Type: "SRv6 OAM (TBA)", Code: "SID not locally implemented (TBA)"); otherwise a success is returned.

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

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

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 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 (see [I-D.draft-filsfils-spring-srv6-network-programming] for details). To solve this problem, the initiator needs to mark the traceroute probe 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 A4::C52, via A2::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 A4::C52, via A2::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 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 END.OTP SID just before the target SID A4::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 (B1::0, A2::C31)(A4::C52, A4::OTP, A2::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 responds 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 END.OTP SID, as described in the pseudocode in Section 3. The packet gets punt 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

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 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-flag 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-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 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.

- 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), 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-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 (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
4.4. In-situ OAM

[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. To address iOAM requirements in an SRv6 network, the draft describes iOAM TLV in SRH and its usage.

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 A100:::2. 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 forward it over link3 with the DA of IPv6 packet set to A4::C52. N4 also performs the normal SRH processing and forward it over link10 with the DA of IPv6 packet set to A100:::. 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. Segment Routing Header Flags Register

This document requests the creation of a new IANA managed registry to identify SRH Flags Bits. The registration procedure is "Expert Review" as defined in [RFC8126]. Suggested registry name is "Segment Routing Header Flags". SRH.Flags is an 8 bits field; the following bit is defined in this document:

<table>
<thead>
<tr>
<th>Suggested Bit</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>OAM</td>
<td>This document</td>
</tr>
</tbody>
</table>

6.2. 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:

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Error</td>
<td>This document</td>
</tr>
<tr>
<td>1</td>
<td>SID is not locally implemented</td>
<td>This document</td>
</tr>
<tr>
<td>2</td>
<td>O-flag punt at Transit</td>
<td>This document</td>
</tr>
</tbody>
</table>

6.3. SRv6 OAM Endpoint Types

This I-D requests to IANA to allocate, within the "SRv6 Endpoint Types" 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:

<table>
<thead>
<tr>
<th>Value (Suggested Value)</th>
<th>Endpoint function</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA (25)</td>
<td>End.OTP</td>
<td>[This.ID]</td>
</tr>
<tr>
<td>TBA (30)</td>
<td>End.OTP</td>
<td>[This.ID]</td>
</tr>
</tbody>
</table>

7. References

7.1. Normative References


7.2. Informative References


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
Abstract

Enhanced VPN (VPN+) is an enhancement to VPN technology to enable it to support the needs of new applications, particularly applications that are associated with 5G services. These applications require better isolation and have more stringent performance requirements than can be provided with overlay VPNs. The characteristics of an enhanced VPN as perceived by its tenant needs to be comparable to those of a dedicated private network. This requires tight integration between the overlay VPN and the underlay network resources in a scalable manner. An enhanced VPN may form the underpin of 5G network slicing, but will also be of use in its own right. This document describes the use of segment routing based mechanisms to provide the enhanced VPN service with dedicated underlay network resources.

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."
Driven largely by needs arising from the 5G mobile network design, the concept of network slicing has gained traction. There is a need to create a VPN with enhanced characteristics. Specifically there is
a need for a transport network supporting a set of virtual networks
each of which provides the client with dedicated (private) network
resources drawn from a shared pool. The tenant of such a network can
require a degree of isolation and performance that previously could
only be satisfied by dedicated networks. Additionally the tenant may
ask for some level of control of their virtual network e.g. to
customize the service paths in the network slice.

The enhanced VPN service (VPN+) as specified in
[I-D.bryant-rtgwg-enhanced-vpn] is targeted at new applications which
require better isolation and have more stringent performance
requirements than can be provided with existing overlay VPNs. An
enhanced VPN may form the underpin of 5G network slicing, but will
also be of use in its own right. Although VPNs can be associated
with dedicated RSVP-TE [RFC3209] LSPs to provide some guarantee to
service performance, such end-to-end per-flow based resource
reservation for each VPN has well-known scalability issues and has
not been the choice in most IP/MPLS networks.

Segment Routing (SR) [I-D.ietf-spring-segment-routing] specifies a
mechanism to steer packet through an ordered list of segments. It
can achieve explicit source routing without introducing per-flow
state into the network. However, currently SR does not have the
capability of resource reservation, it relies on the DiffServ QoS
model to provide coarse-grained service differentiation in the
network. While this may be sufficient for some traditional services,
it cannot meet the requirement of the emerging services.

This document describes the use of segment routing based mechanisms
to provide the enhanced VPN service with dedicated underlay network
resources.

2. Segment Routing with Resource Reservation

In segment routing, several types of segments are defined to
represent either topological elements or service instructions. Node
segment and adjacency segment are two major types of topological
segments. Some other segments may be associated with specific
service functions for service chaining purpose. However, currently
the segments are not associated with network resources for the QoS
purpose.

In order to provide the enhanced VPN for the emerging applications
which require guaranteed performance and isolation from other service
in the network, the overlay VPN needs to been deeply integrated with
underlay network. This requires that some dedicated network
resources need to be allocated exclusively for each enhanced VPN.
When segment routing is used in the network, it is necessary to
introduce resource reservation into segment routing to meet the enhanced VPN requirements.

Following the segment routing paradigm, on each network segment, dedicated network resources are reserved for a particular enhanced VPN. This avoids introducing per-flow state into the network. Specifically, this per-segment resource reservation is achieved by associating the adjacency segments and node segments with resources reserved on the corresponding links and nodes. For example, multiple adjacency segment identifiers (Adj-SIDs) can be allocated for one link, each of which is associated with a set of resource reserved on this link for a particular enhanced VPN, such as link bandwidth, queues, etc. For one particular node, multiple node-SIDs can be allocated for one node, each of which is associated with a set of resource reserved on this node for a particular enhanced VPN. Note that when node-SID is used to build a loose SR path for a particular enhanced VPN, Penultimate Hop Popping (PHP) MUST be disabled, so that every node could use the node-SID to identify the enhanced VPN and the corresponding network resources.

According to the requirement from a particular customer for an enhanced VPN service, the underlay network sub-topology used to support this enhanced VPN is determined. In this sub-topology, the network resources required on each network element are also determined. Network resources are reserved for this enhanced VPN customer in a per-segment manner, and are represented by dedicated node and adjacency SIDs. All the node and adjacency SIDs allocated for this enhanced VPN constitute a virtual SR network, which is used as the underlay of the enhanced VPN service. The extensions to IGP protocol to support the segment routing based resource reservation for enhanced VPN is specified in another document.

3. Procedures

This section describes the detailed procedures of provisioning an enhanced VPN service using segment routing based resource reservation for the enhanced VPN.

Assume that customer A requests an enhanced VPN service from the network operator. The fundamental requirement is that customer A’s traffic does not experience interference from other traffic in the network, such as traffic in other VPNs, or the default traffic in the network. The detailed requirements can be described with following characteristics:

- Service topology
- Service bandwidth
3.1. Topology and Resource Computation

It is assumed that a centralized network controller is responsible for the provisioning of enhanced VPNs. The controller needs to collect the network connectivity, resources and other relevant network state information of the underlay network. This usually can be done using either the IGP [RFC5305] [RFC3630] or BGP-LS [RFC7752].

Based on the network information collected from the elements in the underlay, the controller can compute the best way to meet the requirements of a new tenant whilst maintaining the needs of the existing tenants that are using the same network. The output of the computation is a sub-topology of the underlay network, along with the network resources needed on each network element (e.g. links and nodes) in the sub-topology. This is the underlay network which can fully meet the customer’s service requirement.

3.2. Network Resource and SID Allocation

According to the computation output, the network controller sends requests to all the network devices involved in the sub-topology to allocate the network resources required for this enhanced VPN. This can be done with either PCEP [RFC5440] or Netconf [RFC6241] with necessary extensions. The network resources are allocated in a per-segment manner. Dedicated segment identifiers, e.g. node-SIDs and adj-SIDs are allocated for each network segment along with the network resources reserved for this enhanced VPN.

3.3. Construction of Isolated Virtual Networks

A virtual SR network can then be constructed using the node-SIDs and adj-SIDs allocated for this enhanced VPN. Unlike classical segment routing in which network resources are shared between different services and customers, the proposed mechanism provides a virtual network with dedicated resource reserved in the underlay, so that it can always meet the service requirement of the customer and provide the required isolation from other customers in the same network.

3.4. VPN to Virtual Network Mapping

The services of an enhanced VPN customer would be provisioned using the customized virtual SR networks as the underlay. This ensures that services of different enhanced VPNs will only use the network resources reserved for them and not interfere with each other. For
each enhanced VPN customer, the service paths can be customized for different services within the virtual SR network, and the reserved network resources can be shared by different services of the same enhanced VPN customer.

4. Benefits of SR based Mechanism

The SR based mechanism described in this document provides three things:

- More flexibility and better scaling.
- Lower state maintenance overhead and fewer protocols types in the code than RSVP-TE.
- Better isolation and performance than Classic SR due to allocation of resources in the underlay to specific services.

This SR based mechanism can provide the required isolation between different enhanced VPNs, without introducing per-flow state into the network. For each enhanced VPN, the resource reservation is done in a per-segment manner, which aligns with the segment routing paradigm. This provides better scalability compared to RSVP-TE based per-flow resource reservation.

In addition to isolation, the SR based mechanism also allows resource sharing between different services of the same enhanced VPN customer. This gives the customer more flexibility and control in service planning and provisioning in their dedicated virtual networks, the experience is similar to using a dedicated private network. The performance of critical services in a particular enhanced VPN can be further ensured using the mechanisms defined in [DetNet].

The detailed comparison with other candidates are given in the following sub-sections.

4.1. MPLS-TP

MPLS-TP could be enhanced to include the allocation of specific resources along the path to a specific LSP. This would require that the SDN system set up and maintain every resource at every path for every tenant, and map this to the LSP and hence the unique LSP label at every hop. Whilst this would be a way to produce a proof of concept for network slicing of an MPLS underlay, delegation would be difficult, resulting in a high overhead and high touch system. This leads to scaling concerns. The number of labels needed at any node would be the total number of services passing through that node.
Experience with early pseudowire designs shows that this can lead to scaling issues.

4.2. RSVP-TE

RSVP-TE would have the same scaling concern as MPLS-TP in terms of the number of LSPs that need to be maintained being equal to the number of service passing through any given node. Additionally it would have the two RSVP disadvantages that basic SR seeks to address:

- The use of additional protocol (RSVP) in addition to the routing protocol used to discover the topology and the network resources.
- The overhead of the soft-state maintenance associated with RSVP. The impact of this overhead would be exacerbated by the increased number of end to end paths requiring state maintenance.

4.3. Classic SR

Classic SR minimizes the number of control protocols compared to RSVP to just the routing protocol. It also attempts to minimize the core state by pushing state into the packet. It is not entirely successful with this in that in practise binding SIDs are required to overcome limitations in the ability of some nodes to push large label stacks. Binding SIDs increase network state at strategic places in order to overcome the imposition SID size limit.

In addition, classic SR does not have any resource reservation system below the level of link, and none at node level, which restricts the extent to which some particular tenant traffic can be isolated from other traffic in the network.

4.4. SR with Resource Reservation

The approach described in this document seeks to achieve a compromise between the state limitations of classic TE system and the lack of resource reservation in classic SR.

Specifically, by segmenting the path and allocating resources to virtual network topologies, the operator can choose the granularity of resource to path binding within a topology. In network segments where resource is scarce such that the service requirement cannot be delivered, the SR approach is able to allocate specific resources to a particular service. By contrast, in other parts of the network where resource is plentiful, the resource may be shared by a number of services. The decision to do this is in the hands of the operator or the tenant. Because of the segmented nature of the path, resource aggregation is possible in a way that is not possible with RSVP and
MPLS-TP due to the use of dedicated label to identify each end-to-end path.

5. Service Assurance

In order to provide service assurance it is necessary to instrument the network at multiple levels. The network operator needs to ascertain that the underlay is operating correctly. A tenant needs to ascertain that their services are correctly operating. In principle these can use existing techniques. These are well known problems and solutions either exist or are in development to address them.

However new work is needed to instrument the virtual network slices that are created. Such instrumentation needs to operate without causing disruption to other services using the network. Given the sensitivity of some applications care needs to be taken to ensure that the instrumentation itself does not cause disruption either to the service being instrumented or to another service.

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Security Considerations

The normal security considerations of VPNs are applicable and it is assumed that industry best practise is applied to an enhanced VPN.

The security considerations of segment routing are applicable and it is assumed that these are applied to an enhanced VPN that uses SR.

Some applications of enhanced VPNs are sensitive to packet latency, and the enhanced VPNs provisioned to carry their traffic have latency SLAs. By disrupting the latency of such traffic an attack can be directly targeted at the customer application, or can be targeted at the network operator by causing them to violate their service level agreement and thus causing them commercial consequences. Dynamic attacks of this sort are not something that networks have traditionally guarded against, and networking techniques need to be developed to defend against this type of attack. By rigorously policing ingress traffic and carefully provisioning the resources provided to critical services this type of attack can be prevented. However case needs to be taken where it is necessary to provide
shared resources, and when the network needs to be reconfigured as part of ongoing maintenance or in response to a failure.

It is important that steps are taken to ensure that details of the underlay are not exposed to third parties to minimise the possibility that an exploit be developed as a result of exploiting a shared resource.

8. Acknowledgements

The authors would like to thank Mach Chen, Zhenbin Li for the discussion and suggestions to this document.

9. References

9.1. Normative References


9.2. Informative References


Authors' Addresses

Jie Dong
Huawei Technologies
Email: jie.dong@huawei.com

Stewart Bryant
Huawei Technologies
Email: stewart.bryant@gmail.com

Zhenqiang Li
China Mobile
Email: li_zhenqiang@hotmail.com

Takuya Miyasaka
KDDI Corporation
Email: ta-miyasaka@kddi.com

Abstract

Enhanced VPN (VPN+) is an enhancement to VPN technology to enable it to support the needs of new applications, particularly applications that are associated with 5G services. These applications require better isolation and have more stringent performance requirements than can be provided with overlay VPNs. The characteristics of an enhanced VPN as perceived by its tenant needs to be comparable to those of a dedicated private network. This requires tight integration between the overlay VPN and the underlay network resources in a scalable manner. An enhanced VPN may form the underpin of 5G network slicing, but will also be of use in its own right. This document describes the use of segment routing based mechanisms to provide the enhanced VPN service with dedicated network resources.

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

Copyright Notice

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

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

Table of Contents

1. Introduction ................................................. 3
2. Segment Routing with Resource Allocation ...................... 4
3. Procedures .................................................. 4
   3.1. Topology and Resource Computation ....................... 5
   3.2. Network Resource and SID Allocation ..................... 6
   3.3. Construction of SR Virtual Topology ..................... 7
   3.4. VPN Service to SR Virtual Topology Mapping ............... 8
4. Benefits of the Proposed Mechanism ........................... 8
   4.1. MPLS-TP ................................................. 9
   4.2. RSVP-TE ............................................... 9
   4.3. Classic SR .............................................. 10
   4.4. SR with Resource Allocation .............................. 10
5. Service Assurance ............................................ 10
6. IANA Considerations ......................................... 11
7. Security Considerations ...................................... 11
8. Acknowledgements ............................................ 11
9. References .................................................. 11
   9.1. Normative References .................................... 11
   9.2. Informative References ................................... 12
Authors’ Addresses .............................................. 13
1. Introduction

Driven largely by needs arising from the 5G mobile network design, the concept of network slicing has gained traction. There is a need to create a VPN with enhanced characteristics. Specifically there is a need for a transport network supporting a set of virtual networks each of which provides the client with dedicated (private) network resources drawn from a shared pool. The tenant of such a network can require a degree of isolation and performance that previously could only be satisfied by dedicated networks. Additionally the tenant may ask for some level of control of their virtual network e.g. to customize the service paths in their network slices.

The enhanced VPN service (VPN+) as specified in [I-D.bryant-rtgwg-enhanced-vpn] is targeted at new applications which require better isolation and have more stringent performance requirements than can be provided with existing overlay VPNs. An enhanced VPN may form the underpin of 5G network slicing, but will also be of use in its own right. Although different VPNs can be associated with dedicated RSVP-TE [RFC3209] LSPs to provide some guarantee to the service performance, such mechanisms would introduce per-VPN per-flow states into the network, which is known to have scalability issues and has not been the choice of most IP/MPLS networks.

Segment Routing (SR) [I-D.ietf-spring-segment-routing] specifies a mechanism to steer packet through an ordered list of segments. It can achieve explicit source routing without introducing per-path state into the network. Like RSVP-TE, SR supports source specification of the packet path, However, currently SR does not have the capability of reserving or identifying different network resources for different services or customers. Although the controller can have global view of the network utilization state and can provision different services onto different SR paths, in the data plane it still relies on the DiffServ QoS model to provide coarse-grained traffic differentiation in the network. While this may be sufficient for some traditional services, it cannot meet the requirement of the emerging services.

This document extends the SR paradigm to use segments to identify the different subset of resources allocated on each network elements (links or nodes). The SR Identifiers (SIDs) associated with particular network resources can be used to construct customized virtual networks for different services, the SID can also be used to steer the service traffic to be processed with the corresponding allocated resources. This mechanism can be used to provide the enhanced VPN service with dedicated network resources.
2. Segment Routing with Resource Allocation

In segment routing, several types of segments are defined to represent either topological elements or service instructions. Node segment and adjacency segment are two major types of topological segments. Some other types of segments may be associated with specific service functions for service chaining purpose. However, so far the segments are not associated with network resources for the QoS purpose.

In order to support the enhanced VPNs which require guaranteed performance and isolation from other services in the network, the overlay VPN needs to be integrated with underlay networks. This requires that some dedicated network resources be allocated to the enhanced VPN. When segment routing is used to build enhanced VPNs, it is necessary to associate the segments with network resources.

By extending the segment routing paradigm, different subsets of network resources are allocated from each network element, and associated with different SIDs. On one particular link, multiple adjacency segment identifiers (Adj-SIDs) can be allocated, each of which is associated with a set of resource allocated from the link, such as bandwidth, queues, etc. For one particular node, multiple node-SIDs can be allocated, each of which may be associated with a set of resource allocated from the node, such as the processing resources. This per-segment resource allocation complies to the SR paradigm, which avoids introducing per-path state into the network.

Different groups of SIDs associated with network resources can be used to build the virtual underlay networks for different enhanced VPNs, this provides the required isolation between enhanced VPNs. The adj-SIDs are used to steer traffic of different enhanced VPNs into different link resources on each hop. The node SIDs can be used to steer traffic of different enhanced VPNs into different node resources. The node SIDs can also be used to build loose SR paths for different enhanced VPNs, in which case the node SIDs are used by the transit nodes to steer traffic to use the link resources allocated for the corresponding enhanced VPN. Note that in this case Penultimate Hop Popping (PHP) MUST be disabled, so that each node could use the node-SID to identify the enhanced VPN and the corresponding network resources allocated.

3. Procedures

This section describes the procedures of provisioning an enhanced VPN service based on segment routing with resource allocation.
According to the requirement of an enhanced VPN service, the network controller calculates a sub-topology of the underlay network to support this enhanced VPN. Within the sub-topology, the network resources needed on each network element can also be determined. The network resources are allocated in a per-segment manner, and are associated with different node-SIDs and adj-SIDs. The group of the node-SIDs and adj-SIDs allocated for the enhanced VPN will be used by network nodes and the network controller to build a SR virtual topology, which is used as the virtual underlay of the enhanced VPN service. The extensions to IGP protocol to distribute the SIDs associated with resources allocated for a virtual network topology is specified in [I-D.dong-lsr-sr-enhanced-vpn].

Assume that customer requests for an enhanced VPN service from the network operator. The fundamental requirement is that customer A’s service does not experience interference from other service in the network, such as other VPN services, or the non-VPN services in the network. The detailed requirements can be described using following characteristics:

- Service topology: the service sites and the connectivity between them
- Service bandwidth: the bandwidth requirement between service sites
- Isolation: the level of isolation from other services in the network
- Reliability: whether fast repair or end-to-end protection is needed or not.
- Latency and jitter constraints.

3.1. Topology and Resource Computation

It is assumed that a centralized network controller is responsible for the provisioning of enhanced VPNs. The controller needs to collect the information of network connectivity, network resources, network performance and other relevant network state of the underlay network. This can be done using either the IGP [RFC5305] [RFC3630] [RFC5747] [RFC7810] or BGP-LS [RFC7752] [I-D.ietf-idr-te-pm-bgp].

Based on the network information collected from the underlay network, the controller can compute the best way to meet the requirements of a new tenant whilst maintaining the needs of the existing tenants that are using the same network. The output of the computation is a sub-topology of the underlay network, along with the network resources.
need to be allocated on each network element (e.g. links and nodes) in the sub-topology.

3.2. Network Resource and SID Allocation

According to the computation result of section 3.1, the network controller instructs the network devices involved in the sub-topology of the enhanced VPN to allocate the required network resources. This can be done with either PCEP [RFC5440] or Netconf/YANG [RFC6241] [RFC7950] with necessary extensions. The network resources are allocated in a per-segment manner. In addition, dedicated segment identifiers, e.g. node-SIDs and adj-SIDs are also allocated to represent the network resources allocated for each enhanced VPN on each network segment.

Node-SIDs:
- r:101
- g:201
- b:301

Adj-SIDs:
- r:1001:1G
- g:2001:2G
- b:3001:1G

Figure 1. SID identify resources allocated for different virtual networks

Figure 1 shows a network fragment of enhanced VPN supported by SR. In this example, there are three virtual topologies created for enhanced VPNs red (r), green (g) and blue (b). The red and green topologies consist of nodes A, B, C, D, and E with all their interconnecting links, whilst the blue topology only consists of

nodes A, B, C and D with all their interconnecting links. Each node allocates a dedicated adjacency SID for each link participating in a particular topology. Each node is also allocated with a dedicated node SID for each topology it participates in. The adj-SIDs are associated with the link resources (e.g. bandwidth) allocated for each topology, so that the adj-SIDs can be used to steer service of different enhanced VPNs into different set of reserved resources in the data plane. The node-SIDs can be associated with dedicated nodal resources allocated for each topology. In addition, the node-SIDs of different topologies can be used to build loose SR path within each virtual topology, and steer service of different enhanced VPNs into the different set of reserved resources in the data plane.

In Figure 1, the notation x:nnnn:y that in topology colour x, the adj-SID nnnn will steer the packet over that link which has a total bandwidth of y assigned to that topology. Thus the note r:1002:1G in link C->D says that the red topology over link C->D has a reserved bandwidth of 1Gb/s and will used by a packet arriving at node C with an adj-SID 1002 at the top of the label stack.

3.3. Construction of SR Virtual Topology

Each network node SHOULD advertise the allocated network resources and the associated SIDs for the enhanced VPN into the network. The detailed mechanism and extensions to IGP are described in [I-D.dong-lsr-sr-enhanced-vpn]. The allocated network resources and the associated SIDs for the enhanced VPN needs to be distributed to the network controller for state synchronization and global optimization for each virtual topology. This can be done using either BGP-LS [RFC7752] or Netconf/YANG [RFC6241][RFC7950] with necessary extensions. With the collected network resource and SIDs information, the controller and network nodes are able to construct the SR virtual topologies using the node-SIDs and adj-SIDs allocated for enhanced VPNs. Unlike classical segment routing in which network resources are shared by all services and customers, the SR virtual topologies map to dedicated resource allocated in the underlay, so that they can be used to meet the service requirement of enhanced VPN and provide the required isolation from other services in the same network.

Figure 2 shows the virtual SR topologies created from the underlay network in Figure 1.
3.4. VPN Service to SR Virtual Topology Mapping

The services of an enhanced VPN customer would be provisioned using the customized SR virtual topology as the underlay. This ensures that services of different enhanced VPNs will only use the network resources allocated and will not interfere with each other. For each enhanced VPN customer, the service paths can be customized for different services within the SR virtual topology, and the allocated network resources are shared by different services of the same enhanced VPN customer.

For example, to create a strict path along the path A-B-D-E in the red topology in Figure 2, the SR label stack imposed to the service packet would be (1001, 1002, 1003). For the same strict path in the green topology, the SR label stack would be (2001, 2002, 2003). In the case where we wish to construct a loose path A-D-E in the green topology, the service packet SHOULD be imposed with the SR label stack (201, 204, 205). At node A the packet is sent towards D via either node B or C using the link and node resources allocated for the green topology. At node D the packet is forwarded to E using the link and node resource allocated for the green topology. Similarly, a packet for the loose path A-D-E in the red topology would arrive at node A with the SID stack (101, 104, 105).

4. Benefits of the Proposed Mechanism

Compared with existing mechanisms, The proposed mechanism described in this document provides several key characteristics:

- Flexibility and scalability
- Lower state maintenance overhead and fewer protocols types
- Better isolation and performance than Classic SR due to allocation of resources in the underlay to specific services.

Figure 2. SR virtual topologies using different groups of SIDs
This SR based mechanism can provide the required isolation between different enhanced VPNs, without introducing per-path state into the network. For each enhanced VPN, the resource allocation is done in a per-segment manner, which aligns with the segment routing paradigm. This provides better scalability compared to RSVP-TE based per-flow resource reservation.

In addition to isolation, the SR based mechanism also allows resource sharing between different services of the same enhanced VPN customer. This gives the customer more flexibility and control in service planning and provisioning, the experience would be similar to using a dedicated private network. The performance of critical services flows in a particular enhanced VPN can be further ensured using the mechanisms defined in [DetNet].

The detailed comparison with other candidates technologies are given in the following sections.

4.1. MPLS-TP

MPLS-TP could be enhanced to include the allocation of specific resources along the path to a specific LSP. This would require that the SDN system set up and maintain every resource at every path for every customer, and map this to the LSP in the data plane, hence at every hop unique LSP label is needed for each path. Whilst this would be a way to produce a proof of concept for network slicing of an MPLS underlay, delegation would be difficult, resulting in a high overhead and high touch system. This leads to scaling concerns. The number of labels needed at any node would be the total number of services passing through that node. Experience with early pseudowire designs shows that this can lead to scaling issues.

4.2. RSVP-TE

RSVP-TE would have the same scaling concern as MPLS-TP in terms of the number of LSPs that need to be maintained being equal to the number of service passing through any given node. Additionally it would have the two RSVP disadvantages that basic SR seeks to address:

- The use of additional protocol (RSVP) for path establishment in addition to the routing protocol used to discover the topology and the network resources.

- The overhead of the soft-state maintenance associated with RSVP. The impact of this overhead would be exacerbated by the increased number of end to end paths requiring state maintenance.
4.3. Classic SR

Classic SR minimizes the number of control protocols compared to RSVP-TE to just the routing protocol. It also attempts to minimize the core state by pushing state into the packet, although limited number of binding SIDs may be required to overcome the limitations in the ability of some nodes to push large label stacks. However, classic SR does not have any resource allocation and identification mechanism below the level of link, and none at node level, which restricts the extent to which some particular tenant traffic can be isolated from other traffic in the network.

4.4. SR with Resource Allocation

The approach described in this document seeks to achieve a compromise between the state limitations of classic TE system and the lack of resource awareness in classic SR.

Specifically, by segmenting the path and allocating network resources to each element of the virtual network topologies, the operator can choose the granularity of resource to path binding within a virtual topology. In network segments where resource is scarce such that the service requirement cannot be delivered, the SR approach is able to allocate specific resources to a particular service. By contrast, in other parts of the network where resource is plentiful, the resource may be shared by a number of services. The decision to do this is in the hands of the operator. Because of the segmented nature of the path, resource aggregation is possible in a way that is not possible with RSVP-TE and MPLS-TP due to the use of dedicated label to identify each end-to-end path.

5. Service Assurance

In order to provide service assurance it is necessary to instrument the network at multiple levels. The network operator needs to ascertain that the underlay is operating correctly. A tenant needs to ascertain that their services are correctly operating. In principle these can use existing techniques. These are well known problems and solutions either exist or are in development to address them.

However new work is needed to instrument the virtual network topologies that are created. Such instrumentation needs to operate without causing disruption to other services using the network. Given the sensitivity of some applications care needs to be taken to ensure that the instrumentation itself does not cause disruption either to the service being instrumented or to another service.
6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Security Considerations

The normal security considerations of VPNs are applicable and it is assumed that industry best practise is applied to an enhanced VPN.

The security considerations of segment routing are applicable and it is assumed that these are applied to an enhanced VPN that uses SR.

Some applications of enhanced VPNs are sensitive to packet latency, and the enhanced VPNs provisioned to carry their traffic have latency SLAs. By disrupting the latency of such traffic an attack can be directly targeted at the customer application, or can be targeted at the network operator by causing them to violate their service level agreement and thus causing them commercial consequences. Dynamic attacks of this sort are not something that networks have traditionally guarded against, and networking techniques need to be developed to defend against this type of attack. By rigorously policing ingress traffic and carefully provisioning the resources provided to critical services this type of attack can be prevented. However case needs to be taken where it is necessary to provide shared resources, and when the network needs to be reconfigured as part of ongoing maintenance or in response to a failure.

It is important that steps are taken to ensure that details of the underlay are not exposed to third parties to minimise the possibility that an exploit be developed as a result of exploiting a shared resource.

8. Acknowledgements

The authors would like to thank Mach Chen, Zhenbin Li for the discussion and suggestions to this document.

9. References

9.1. Normative References
9.2. Informative References


Authors' Addresses

Jie Dong
Huawei Technologies

Email: jie.dong@huawei.com

Stewart Bryant
Huawei Technologies

Email: stewart.bryant@gmail.com
Zhenqiang Li
China Mobile
Email: li_zhenqiang@hotmail.com

Takuya Miyasaka
KDDI Corporation
Email: ta-miyasaka@kddi.com
Abstract

Segment Routing 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 the 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

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

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

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

This Internet-Draft will expire on September 1, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction .............................................. 5
2. SR Policy ................................................ 5
   2.1. Identification of an SR Policy ....................... 5
   2.2. Candidate Path and Segment List ..................... 6
   2.3. Protocol-Origin of a Candidate Path .................. 6
   2.4. Originator of a Candidate Path ....................... 7
   2.5. Discriminator of a Candidate Path ................... 7
   2.6. Identification of a Candidate Path ................. 8
   2.7. Preference of a Candidate Path ....................... 8
   2.8. Validity of a Candidate Path ......................... 8
   2.9. Active Candidate Path ................................ 8
   2.10. Validity of an SR Policy ........................... 10
   2.11. Instantiation of an SR Policy in the Forwarding Plane .. 10
   2.12. Priority of an SR Policy ........................... 10
2.13. Summary ............................................. 10
3. Segment Routing Database .............................. 11
4. Segment Types ........................................... 12
  4.1. Explicit Null ...................................... 15
5. Validity of a Candidate Path ......................... 15
  5.1. Explicit Candidate Path ........................... 16
  5.2. Dynamic Candidate Path ........................... 17
6. Binding SID ............................................. 17
  6.1. BSID of a candidate path ......................... 17
  6.2. BSID of an SR Policy ............................. 17
  6.2.1. Frequent use-cases: unspecified BSID .......... 18
  6.2.2. Frequent use-case: all specified to the same BSID 18
  6.2.3. Specified-BSID-only .......................... 18
  6.3. Forwarding Plane .................................. 18
  6.4. Not an identification ............................ 19
7. SR Policy State ........................................ 19
8. Steering into an SR Policy ........................... 19
  8.1. Validity of an SR Policy ......................... 19
  8.2. Drop upon invalid SR Policy ..................... 20
  8.3. Incoming Active SID is a BSID .................. 20
  8.4. Per-Destination Steering ......................... 21
  8.4.1. Multiple Colors ............................... 21
  8.5. Recursion on an on-demand dynamic BSID .......... 22
  8.5.1. Multiple Colors ............................... 22
  8.6. Per-Flow Steering ................................ 22
  8.7. Policy-based Routing ............................. 23
  8.8. Optional Steering Modes for BGP Destinations ... 24
  8.8.1. Color-Only BGP Destination Steering .......... 24
  8.8.2. Multiple Colors and CO flags ................. 25
  8.8.3. Drop upon Invalid ............................ 25
9. Other type of SR Policies ............................ 26
  9.1. Layer 2 and Optical Transport ................... 26
  9.2. Spray SR Policy ................................... 27
10. 50msec Local Protection ............................ 27
  10.1. Leveraging TI-LFA local protection of the constituent IGP segments ........................... 27
  10.2. Using an SR Policy to locally protect a link .. 28
11. Other types of Segments .............................. 28
  11.1. Service SID ...................................... 28
  11.2. Flex-Alg IGP SID ................................ 29
12. Binding SID to a tunnel .............................. 29
13. Traffic Accounting .................................... 29
  13.1. Traffic Counters Naming convention ............ 30
  13.2. Per-Interface SR Counters ....................... 31
  13.2.1. Per interface, per protocol aggregate egress SR traffic counters (SR.INT.E.PRO) ......... 31
  13.2.2. Per interface, per traffic-class, per protocol aggregate egress SR traffic counters
13.2.3. Per interface aggregate ingress SR traffic counter
(SR.INT.I) ........................................ 31
13.2.4. Per interface, per TC aggregate ingress SR traffic
counter (SR.INT.I.TC) ............................. 32
13.3. Prefix SID Counters ................................ 32
13.3.1. Per-prefix SID egress traffic counter (PSID.E) ... 32
13.3.2. Per-prefix SID per-TC egress traffic counter
(PSID.E.TC) ........................................... 32
13.3.3. Per-prefix SID, per egress interface traffic counter
(PSID.INT.E) ........................................ 32
13.3.4. Per-prefix SID per TC per egress interface traffic
counter (PSID.INT.E.TC) ........................... 32
13.3.5. Per-prefix SID, per ingress interface traffic
counter (PSID.INT.I) ............................... 33
13.3.6. Per-prefix SID, per TC, per ingress interface
traffic counter (PSID.INT.I.TC) ................. 33
13.4. Traffic Matrix Counters ............................ 33
13.4.1. Per-Prefix SID Traffic Matrix counter (PSID.E.TM) 33
13.4.2. Per-Prefix, Per TC SID Traffic Matrix counter
(PSID.E.TM.TC) ..................................... 33
13.5. SR Policy Counters .................................. 34
13.5.1. Per-SR Policy Aggregate traffic counter (POL) ... 34
13.5.2. Per-SR Policy labelled steered aggregate traffic
counter (POL.BSID) ................................. 34
13.5.3. Per-SR Policy, per TC Aggregate traffic counter
(POL.TC) ............................................. 34
13.5.4. Per-SR Policy, per TC labelled steered aggregate
traffic counter (POL.BSID.TC) ..................... 34
13.5.5. Per-SR Policy, Per-Segment-List Aggregate traffic
counter (POL.SL) .................................... 35
13.5.6. Per-SR Policy, Per-Segment-List labelled steered
aggregate traffic counter (POL.SL.BSID) ......... 35
14. Appendix A ............................................. 35
14.1. SRTE headend architecture .......................... 35
14.2. Distributed and/or Centralized Control Plane .......... 36
14.2.1. Distributed Control Plane within a single Link-State
IGP area ............................................. 36
14.2.2. Distributed Control Plane across several Link-State
IGP areas ............................................ 36
14.2.3. Centralized Control Plane .......................... 37
14.2.4. Distributed and Centralized Control Plane .......... 37
14.3. Examples of Candidate Path Selection ................ 38
14.4. More on Dynamic Path .................................. 41
14.4.1. Optimization Objective ............................ 41
14.4.2. Constraints ....................................... 42
14.4.3. SR Native Algorithm ............................... 42
14.4.4. Path to SID .................................... 43
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 [I-D.ietf-spring-segment-routing].

The headend node is said to steer a flow into an Segment Routing Policy (SR Policy).

The header of a packet steered in an SR Policy is augmented with the ordered list of segments associated with that SR Policy.

This document details the concepts of SR Policy and steering 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

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.

The endpoint indicates the destination of the policy. The endpoint is specified as an IPv4 or IPv6 address. 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).
2.2. Candidate Path and Segment List

An SR Policy is associated with one or more candidate paths.

A candidate path is itself associated with a Segment-List (SID-List) or a set of SID-Lists. In the latter case, each SID-List is associated with a weight for weighted load balancing (refer to section 2.11 for details). The default weight is 1.

A SID-List represents a specific source-routed way to send traffic from the head-end to the endpoint of the corresponding SR policy.

A candidate path is either dynamic or explicit.

An explicit candidate path is associated with a SID-List or a set of SID-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 SID-List (or set of SID-Lists) that solves the optimization problem.

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 [I-D.ietf-pce-pce-initiated-lsp] or BGP [I-D.draft-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. Implementations MAY allow modifications of these default values assigned to protocols on the SRTE head-end as long as no two protocols share the same value.

The default values are listed below:

<table>
<thead>
<tr>
<th>Value</th>
<th>Protocol-Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>PCEP</td>
</tr>
<tr>
<td>20</td>
<td>BGP-SRTE</td>
</tr>
<tr>
<td>30</td>
<td>Local (via CLI, Yang model through NETCONF, gRPC, etc.)</td>
</tr>
</tbody>
</table>

Table 1: Protocol-origin Identifier
2.4. Originator of a Candidate Path

Originator identifies the node which provisioned or signalled the candidate path on the SRTE head-end. 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.

When Protocol-Origin is Local, the ASN and node address MAY be set to either the SRTE 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-SRTE, 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 Local, this is an implementation’s configuration model specific unique identifier for a candidate path.

When PCEP is the Protocol-Origin, the method to uniquely identify signalled path will be specified in an upcoming PCEP draft.
When BGP-SRTE is the Protocol-Origin, it is the distinguisher specified in Section 2.1 of [I.D.draft-ietf-idr-segment-routing-te-policy].

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 SID-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 SID-List. The SID-List does not identify a candidate path. The SID-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 identify 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 valid if it is usable. 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. Lower value of originator is selected.
3. Finally, the higher value of discriminator is selected.

An implementation MAY choose to override any of the tie-breaking rules above and maintain the already selected candidate path as active path.

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 locally provisioned over paths signalled via BGP-SRTE 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-SRTE over what is locally provisioned.

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.
Section 14.3 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 is used for forwarding traffic that is being steered onto that policy.

If a set of SID-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 SID-List.

The fraction of the flows associated with a given SID-List is \( w/Sw \) where \( w \) is the weight of the SID-List and \( Sw \) is the sum of the weights of the SID-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. An implementation MAY provide a per-policy priority field. 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.

2.13. Summary

In summary, the information model is the following:

```plaintext
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
```

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 SID-Lists of CP1 are installed as the forwarding instantiation of SR policy POL1. Traffic steered on POL1 is flow-based hashed on SID-List <SID1...SID1i> with a ratio W1/(W1+W2).

3. Segment Routing Database

An SR headend maintains the Segment Routing Traffic Engineering Database (SRTE-DB).

An SR headend leverages the SRTE-DB to validate explicit candidate paths and compute dynamic candidate paths.

The information in the SRTE-DB MAY include:

- IGP information (topology, IGP metrics).
- 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-Domain Topology information [I.D.draft-ietf-idr-bgpls-segment-routing-epe].
- Segment Routing information (such as SRGB, SRLB, Prefix-SIDs, Adj-SIDs, BGP Peering SID, SRv6 SIDs).

The SRTE-DB is multi-domain capable.

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 SRTE-DB may only contain the attached domain topology while in others, the SRTE-DB may contain the topology of multiple domains. The SRTE-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 ([I-D.ietf-pce-stateful-pce] 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 (Section 6).

The SRTE-DB MAY also contain the Maximum SID Depth (MSD) capability of nodes in the topology. This can be collected via ISIS [draft-ietf-isis-segment-routing-msd], OSPF [draft-ietf-ospf-segment-routing-msd], BGP-LS [draft-ietf-idr-bgp-ls-segment-routing-msd] or PCEP [I-D.ietf-pce-segment-routing].

4. Segment Types

A SID-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 SID-List. However, the SID-List itself can specified using different segment-descriptor types and the following are defined:

Type 1: SR-MPLS Label:
SR-MPLS label corresponding to any of the segment types defined in [I.D.draft-ietf-spring-segment-routing] 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 SRTE process on the headend to perform any resolution.

Type 2: SRv6 SID:
IPv6 address corresponding to any of the segment types defined in [I.D.draft-filsfils-spring-srv6-network-programming] can be used. This type does not require the SRTE process on the headend to perform any resolution.

Type 3: IPv4 Prefix with optional SR Algorithm:
The SRTE process on the headend is required to resolve the specified IPv4 Prefix Address to the SR-MPLS label corresponding to its Prefix SID segment. The SR algorithm (refer to Section 3.1.1 of [I.D.draft-ietf-spring-segment-routing]) to be used MAY also be provided. When algorithm is not specified, the SRTE process is expected to use the Prefix SID signalled for the Strict Shortest Path algorithm when available and if not then use the Shortest Path or default algorithm.
Type 4: IPv6 Global Prefix with optional SR Algorithm for SR-MPLS:
In this case the SRTE process on the headend is required to resolve the specified IPv6 Global Prefix Address to the SR-MPLS label corresponding to its Prefix SID segment. The SR Algorithm (refer to Section 3.1.1 of [I.D.draft-ietf-spring-segment-routing]) to be used MAY also be provided. When algorithm is not specified, the SRTE process is expected to use the Prefix SID signalled for the Strict Shortest Path algorithm when available and if not then use the Shortest Path or default algorithm.

Type 5: IPv4 Prefix with Local Interface ID:
This type allows identification of Adjacency SID or BGP EPE Peer Adjacency SID label for point-to-point links including IP unnumbered links. The SRTE process on 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 for BGP EPE Peer Adjacency SID label for links. The SRTE process on 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 or BGP EPE Peer Adjacency SID label for links including those with only Link Local IPv6 addresses. The SRTE process on 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 for BGP EPE Peer Adjacency SID label for links with Global IPv6 addresses. The SRTE process on 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 SRTE process on the headend is required to resolve the specified IPv6 Global Prefix Address to the SRv6 END function SID corresponding to the node which is originating the prefix. The SR Algorithm (refer to Section 3.1.1 of [I.D.draft-ietf-spring-segment-routing]) to be used MAY also be provided. When algorithm is not specified, the SRTE process is expected to use the Prefix SID signaled for the Strict Shortest Path algorithm when available and if not then use the Shortest Path or default algorithm.

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 for links with only Link Local IPv6 addresses. The SRTE process on 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 for links with Global IPv6 addresses. The SRTE process on 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 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 first segment the traffic will be directed toward along the SR explicit path.
- The last Segment represents the bottommost label or the last IPv6 segment the traffic will be directed toward along the SR explicit 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 SID-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 SID-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 SID-List(s) based on IPv4 SIDs, the Explicit Null Label Policy is processed as specified in draft-idr-segment-routing-te-policy Section 2.4.4. When this is not present then the headend SHOULD automatically impose the "IPv6 Explicit NULL Label" as bottom of stack label. Refer to "Steering" section later in this document.

5. Validity of a Candidate Path
5.1. Explicit Candidate Path

An explicit candidate path is associated with a SID-List or a set of SID-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 SID list is external to the SR Policy headend. The SR Policy headend does not compute the SID list. The SR Policy headend only confirms its validity.

A SID-List of an explicit candidate path MUST be declared invalid when:

- It is empty.
- Its weight is 0.
- The headend is unable to resolve the first SID into one or more outgoing interface(s) and next-hop(s).
- The headend is unable to resolve any non-first SID of type 3-to-11 into an MPLS label or an SRv6 SID.

"Unable to resolve" means that the headend has no path to the SID in its SRTE-DB.

In multi-domain deployments, it is expected that the headend be unable to verify the reachability of the SIDs in remote domains. Types 1 and 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 SID-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 SID-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 SRTE-DB to compute a SID-List ("solution SID-List") that solves this optimization problem.

The headend re-computes the solution SID-List any time the inputs to the problem change (e.g., topology changes).

When local computation is not possible (e.g., a policy’s tail-end is outside the topology known to the head-end) or not desired, the head-end 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 is declared invalid.

Section 14.4 lists 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 SID list.

6. Binding SID

The Binding SID (BSID) is fundamental to Segment Routing [I.D.draft-ietf-spring-segment-routing]. It provides scaling, network opacity and service independence. Section 14.5 illustrates these benefits.

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., SRLB).

When the specified BSID is not available (optionally is not in the SRLB), an alert message is 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 $B_1$, 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, then the SR Policy keeps the previous BSID $B_1$.

6.2.1. Frequent use-cases: unspecified BSID

All the candidate paths of the same SR Policy have 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 path.

6.2.2. Frequent use-case: all specified to the same BSID

All the paths of the SR Policy have the same specified BSID.

6.2.3. Specified-BSID-only

A headend MAY be configured with the Specified-BSID-only restrictive behavior.

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 is triggered. Other candidate paths can 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. Not an identification

The association of an SR Policy to a BSID MAY change over the life of
the SR policy (e.g., upon active path change). The BSID of an SR
Policy is not an identification of an SR policy. The identification
of an SR Policy is the tuple <headend, color, endpoint>.

7. SR Policy State

The SR Policy State is maintained on the headend by the SRTE process
represents the state of the policy and its candidate paths to provide
the accurate representation of whether the policy is being
instantiated in the forwarding plane and which of the candidate paths
is 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.

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
  head-end.
- 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.

By default, upon transitioning to the invalid state,
8.2. Drop upon invalid SR Policy

An SR Policy MAY be enabled for the Drop-Upon-Invalid behavior:

- any 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 SID-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 in 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 (endpoint = N, color = C) of SID-List <S1, S2, S3> and BSID B.
- has a BGP policy which matches on the extended-color community C and allows its usage as an SRTE 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 SRTE path to 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 SID-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 ([ID.draft-ietf-idr-tunnel-encaps-07], [I.D.draft-ietf-idr-segment-routing-te-policy]), any AFI/SAFI of LISP [RFC6830].

8.4.1. Multiple Colors

When a BGP route has multiple extended-color communities each with a valid SRTE policy, the BGP process installs the route on the SR policy whose color is of highest numerical value.

Let us assume that headend H:
o learns a BGP route R/r via next-hop N, extended-color communities C1 and C2 and VPN label V.

o has a valid SR Policy P1 to (endpoint = N, color = C1) of SID list <S1, S2, S3> and BSID B1.

o has a valid SR Policy P2 to (endpoint = N, color = C2) of SID list <S4, S5, S6> and BSID B2.

o has a BGP policy which matches on the extended-color communities C1 and C2 and allows their usage as an SRTE 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 (endpoint N, color C).

In this section, independently to the a-priori existence of any explicit candidate path of the SR policy (N, C), it is to be noted that the BGP process at node H triggers the SRTE process at node H to instantiate a dynamic candidate path for the SR policy (N, C) as soon as:

o the BGP process learns of a route R/r via N and with color C.

o a local policy at node H authorizes the on-demand SRTE path instantiation and maps the color to a dynamic SRTE 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 SRTE dynamic path request (endpoint N, color Ci) is triggered for each color Ci.

8.6. Per-Flow Steering

Let us assume that head-end H:

o has a valid SR Policy P1 to (endpoint = N, color = C1) of SID-List <S1, S2, S3> and BSID B1.

o has a valid SR Policy P2 to (endpoint = N, color = C2) of SID-List <S4, S5, S6> and BSID B2.

o 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.

This realizes per-flow steering: different flows bound to the same
BGP endpoint are steered on different IGP or SRTE paths.

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/v6 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 IPv4/6 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 IPv4/6
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 (endpoint = N, color = C) of SID-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 an SRTE 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 10.2 applied to BGP-based steering.

9. Other type of SR Policies

9.1. Layer 2 and Optical Transport

\[
\begin{array}{c}
1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \\
I2(\text{lambda L241}) \quad / \quad I4(\text{lambda L241}) \\
\text{Optical}
\end{array}
\]

Figure 1: 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 1 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 SID list <I2(L241)> and Binding SID B where I2(L241) is of type 5 (IPv4) or type 7 (IPv6), see section 4.
- node 1 steers a packet P1 towards the prefix SID of node 5 (16005).
- node 1 steers a packet P2 on the SR policy <16002, B, 16005>. 

Filsfils, et al. Expires September 1, 2018
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 SRTE 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 1, the DWDM characteristics of the SR Policy at node 2 in terms of latency, loss, security, domain/country traversed by the circuit etc.).

9.2. Spray SR Policy

A Spray SRTE policy is a variant of an SRTE policy which involves packet replication.

Any traffic steered into a Spray SR Policy is replicated along the SID-Lists of its selected path.

In the context of a Spray SR Policy, the selected path SHOULD have more than one SID-List. The weights of the SID-Lists is not applicable for a Spray SR Policy. They MUST be set to 1.

Like any SR policy, a Spray SR Policy has a BSID instantiated into the forwarding plane.

Traffic is typically steered into a Spray SR Policy in two ways:

- local policy-based routing at the headend of the policy.
- remote classification and steering via the BSID of the Spray SR Policy.

10. 50msec Local Protection

10.1. Leveraging TI-LFA local protection of the constituent IGP segments

In any topology, Topology-Independent LFA (TI-LFA) [I.D.draft-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 leverage 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.

10.2. Using an SR Policy to locally protect a link

```
1----2-----6----7
 |    |     |    |
4----3-----9----8
```

Figure 2: Local protection using SR Policy

An SR Policy can be instantiated at node 2 to protect the link 2to6.
A typical explicit SID 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.

11. Other types of Segments

The Segment Routing architecture specifies that any instruction can be bound to a segment.

Similarly, an SR Policy can be composed of SIDs of any types.

On top of the classic IGP SIDs, BGP SIDs and BSIDs, this section highlights the use of service SIDs and IGP-Flex-Alg SIDs.

11.1. Service SID

A Service Segment is a Segment associated with a service, either directly or via an SR proxy. A service may be 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 [I.D.draft-clad-spring-segment-routing-service-chaining].

An SR Policy can be composed of a mix of segments of various types: IGP segments, BGP segments, Binding SIDs and Service Segments.
Similarly to other segments, service segments can be discovered via BGP-LS [I.D. draft-dawra-idr-bgp-sr-service-chaining].

11.2. Flex-Alg IGP SID

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 FlexAlgl
- SID(k, 0) represents the PrefixSID of node k according to Alg=0.
- SID(k, FlexAlgl) 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, FlexAlgl), SID9(FlexAlgl)>.

12. Binding SID to a tunnel

A Binding SID can be bound to any type of tunnel: IP tunnel, GRE tunnel, IP/UDP tunnel, MPLS RSVP-TE tunnel, etc.

13. Traffic Accounting

This section describes counters for traffic accounting in 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 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. The traffic counters are divided into four categories: interface counters, prefix counters,
counters to measure the traffic (demand) matrix and SR policy counters at the policy head-end.

13.1. 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.
13.2. 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.

13.2.1. 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).

13.2.2. 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).

13.2.3. 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.
13.2.4. 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.

13.3. 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.

13.3.1. 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.

13.3.2. 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.

13.3.3. 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.

13.3.4. 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.

13.3.5. 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.

13.3.6. 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.

13.4. Traffic Matrix Counters

A Traffic Matrix (TM) 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.

13.4.1. 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.

13.4.2. 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.

13.5. SR Policy Counters

Per policy counters are only maintained at the policy head-end node. For each SR policy, the head-end node maintains the following counters.

13.5.1. 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.

13.5.2. 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.

13.5.3. 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.

13.5.4. 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.
13.5.5. 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.

13.5.6. 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.

14. Appendix A

14.1. SRTE headend architecture

```
+--------+  +--------+
|  BGP   |  |  PCEP  |
+--------+  +--------+
    \   /   \
    +--------+  +--------+  +--------+
    |  CLI  |  |  SRTE  |  |  NETCONF|
    +--------+  +--------+  +--------+
        \   /     \
        +--------+     +--------+
        |  FIB  |        |
        +--------+
```

**Figure 4: SRTE Architecture at a Headend**

The SRTE functionality at a headend can be implemented in an SRTE process as illustrated in Figure 1.

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

The SRTE process selects the active path of an SR Policy.
The SRTE 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 SRTE process maintains an SRTE-DB. The SRTE process interacts with other processes (Figure 2) to collect the SRTE-DB information.

```
+--------+  +--------+
| BGP-LS |  | IGP   |
+--------+  +--------+
    \   /    \\
   +--------+  +--------+  +--------+
   | PCEP | --| SRTE | --| NETCONF|
   +--------+  +--------+  +--------+
```

Figure 5: Topology/link-state database architecture

The SRTE architecture supports both centralized and distributed control-plane.

14.2. Distributed and/or Centralized Control Plane

14.2.1. 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 SRTE process at H learns the topology (and extended TE latency information) from the IGP and computes the solution SID list providing the low-latency path to E.

No centralized controller is involved in such a deployment.

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

14.2.2. 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 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.

Thanks to a BGP-LS session to any BGP-LS RR, H’s SRTE 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 SID list that
spans the two domains I1 and I2.

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

No centralized controller is required.

### 14.2.3. Centralized Control Plane

Considering the same domain D as in the previous section, let us know
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 SID list \(<S1, S2, S3>\) and programs H with the
related explicit candidate path.

The headend H does not compute the solution SID list (it cannot).
The headend only validates the received explicit candidate path.
Most probably, the controller encodes the SID’s of the SID-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 SRTE-DB at H only uses the LSDB provided by the IGP I1.

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

### 14.2.4. Distributed and Centralized Control Plane

Consider the same domain D as in the previous section.

H’s SRTE 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 SID list (e.g., PCEP).

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 SID list (entirely distributed when the endpoint is in the SRTE DB of the headend) else delegates the computation to a controller (hybrid distributed/centralized control-plane).

The SRTE-DB at H only uses the LSDB provided by the IGP.

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

14.3. Examples of Candidate Path Selection

Example 1:

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

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

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

o Because the NLRIs are identical (same distinguisher), BGP will perform bestpath selection. Note that there are no changes to BGP best path selection algorithm.

o H installs one advertisement as bestpath into the BGP table.

o A single advertisement is passed to the SRTE process.

o SRTE 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 SRTE process.
- SRTE 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 (endpoint, color) 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 SRTE process.
- SRTE process at H selects the candidate path indicated by "foo" as the active path for the SR policy since P2 is greater than P1.

When an SR Policy has multiple valid candidate paths with the same best preference, the SRTE process at a headend uses the rules described in section 2.9 to select the active path as 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 both 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 SRTE process at time t1).

After t1, SRTE process at H selects candidate path associated with NLRI B as active path of the SR policy since RD2 is higher than RD1.
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 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 SRTE process at time t1).

Even after t1, SRTE 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 SRTE process at time t1).

Even after t1, SRTE 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 SRTE process at time t1).

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

14.4. More on Dynamic Path

14.4.1. Optimization Objective

This document defines two optimization objectives:

- Min-Metric - requests computation of a solution SID-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 SID-List.

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

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

When the selected O metric is not the IGP metric, then the solution SID-List is made of prefix SIDs of intermediate nodes, Adjacency SIDs along intermediate links and potentially 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 SID-List such that packets flowing through the solution SID-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 the solution SID-List minimizes the O metric while meeting the maximum number of SID constraints.

14.4.2. Constraints

The following constraints can be defined:

- 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 SID-List.
- Maximum number of weighted SID-Lists in the solution set.
- Diversity to another service instance (e.g., link, node, or SRLG disjoint paths originating from different head-ends).

14.4.3. 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 SID-List. The solution SID-List would be <4, 5, 7, 3>.

An SR-native algorithm would find a SID-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 SID-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 to reflect the TDM paradigm).

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

14.4.4. 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).

```
A---D
 |   |
 B---C
```

Figure 7: 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.
14.5. Benefits of Binding SID

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

A---DCI1----C-----D----E----DCI3---H
\                     \         /   \
S |         |         | Z
\                     \         /   \
B---DCI2----F---------G----DCI4---K
<==DC1==><=========Core========><==DC2==>

Figure 8: A Simple Datacenter Topology

A simplified illustration is provided on the basis of the previous diagram 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).

In this example, it is assumed no redistribution between the IGP’s and no presence of BGP. 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.

14.6. Centralized Discovery of available SID in SRLB

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 Segment Routing Label Block (SRLB).
- its local topology.
- its topology-related SIDs (Adj SID and EPE SID).
- its SR Policies and their BSID ([I-D.ietf-idr-te-lsp-distribution]).

Any controller can thus infer the available SIDs in the SRLB of any node.

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, 5000] 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 ([I-D.ietf-idr-te-lsp-distribution]) or PCEP notification. 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.

15. Acknowledgement

The authors like to thank Tarek Saad and Dhanendra Jain for their valuable comments and suggestions.

16. Normative References

[GLOBECOM]

[I-D.ietf-idr-te-lsp-distribution]

[I-D.ietf-isis-segment-routing-extensions]
[I-D.ietf-pce-pce-initiated-lsp]

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

[I-D.ietf-pce-stateful-pce]

[I-D.ietf-spring-segment-routing]

[I-D.previdi-idr-segment-routing-te-policy]

[I-D.sivabalan-pce-binding-label-sid]


Authors’ Addresses

Clarence Filsfils  
Cisco Systems, Inc.  
Pegasus Parc  
De kleetlaan 6a, DIEGEM  BRABANT 1831  
BELGIUM  
Email: cfilsfil@cisco.com

Siva Sivabalan  
Cisco Systems, Inc.  
2000 Innovation Drive  
Kanata, Ontario  K2K 3E8  
Canada  
Email: msiva@cisco.com

Kamran Raza  
Cisco Systems, Inc.  
2000 Innovation Drive  
Kanata, Ontario  K2K 3E8  
Canada  
Email: skraza@cisco.com

Jose Liste  
Cisco Systems, Inc.  
821 Alder Drive  
Milpitas, California  95035  
USA  
Email: jliste@cisco.com

Francois Clad  
Cisco Systems, Inc.  
Email: fclad@cisco.com

Ketan Talaulikar  
Cisco Systems, Inc.  
Email: ketant@cisco.com
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 the 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

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 November 22, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction ................................................. 3
2. SR Policy .............................................. 4
   2.1. Identification of an SR Policy ...................... 4
   2.2. Candidate Path and Segment List ................... 5
   2.3. Protocol-Origin of a Candidate Path ............... 5
   2.4. Originator of a Candidate Path .................... 6
   2.5. Discriminator of a Candidate Path ................. 7
   2.6. Identification of a Candidate Path ............... 7
   2.7. Preference of a Candidate Path .................... 8
   2.8. Validity of a Candidate Path ...................... 8
   2.9. Active Candidate Path ............................. 8
   2.10. Validity of an SR Policy ......................... 9
   2.11. Instantiation of an SR Policy in the Forwarding Plane 9
   2.12. Priority of an SR Policy .......................... 10
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 [I-D.ietf-spring-segment-routing].

The headend node is said to steer a flow into an Segment Routing Policy (SR Policy).

The header of a packet steered in an SR Policy is augmented with the ordered list of segments associated with that SR Policy.
This document details the concepts of SR Policy and steering 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 [I-D.ietf-spring-segment-routing] specifies that any instruction can be bound to a segment. Thus, an SR Policy can be built using any types of Segment Identifiers (SIDs) 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.

The endpoint indicates the destination of the policy. The endpoint is specified as an IPv4 or IPv6 address. 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 an user-friendly attribute for debug and troubleshooting purposes. Such symbolic names MUST NOT be considered as identifiers for an SR Policy.
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 candidate path is itself associated with a Segment-List (SID-List) or a set of SID-Lists.

A SID-List represents a specific source-routed way 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 associated with a SID-List or a set of SID-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 SID-List (or set of SID-Lists) that solves the optimization problem.

When a candidate path is associated with a set of SID-Lists, each SID-List is associated with a weight for weighted load balancing (refer Section 2.11 for details). The default weight is 1.

A variation of SR Policy can be used for packet replication. A candidate path could comprise multiple SID-Lists; one for each replication path. In such a scenario, packets are actually replicated through each SID List of the SR Policy to realize a point-to-multipoint service delivery. The weight of each SID-List does not come into the picture in this case since there is no load-balancing. The details of this and other such mechanisms for use of SR Policy for point-to-multipoint delivery are outside the scope of this document.

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  | Local (via CLI, Yang model through NETCONF, gRPC, etc.) |

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 Local, 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
o 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

o 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 Local, 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 SID-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 SID-List. The SID-List does not identify a candidate path. The SID-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 identify 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 valid if it is usable. 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. Lower value of originator is selected.
3. Finally, the higher value of discriminator is selected.
An implementation MAY choose to override any of the tie-breaking rules above and maintain the already selected candidate path as active path.

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 locally provisioned 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 locally provisioned.

- 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 is used for forwarding traffic that is being steered onto that policy.
If a set of SID-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 SID-List.

The fraction of the flows associated with a given SID-List is \( \frac{w}{S_w} \) where \( w \) is the weight of the SID-List and \( S_w \) is the sum of the weights of the SID-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. 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 <SID1...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 SID-Lists of CP1 are installed as the...
forwarding instantiation of SR policy Pol1. Traffic steered on Pol1 is flow-based hashed on SID-List <SID11...SID1i> with a ratio W1/(W1+W2).

3. Segment Routing Database

An SR headend maintains the Segment Routing Database (SR-DB).

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) [I-D.ietf-spring-segment-routing]
  [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
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 SID-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 SID-List. However, the SID-List itself can 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 [I-D.ietf-spring-segment-routing] 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 [I-D.ietf-spring-segment-routing]). The SR algorithm (refer to Section 3.1.1 of [I-D.ietf-spring-segment-routing]) 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 [I-D.ietf-spring-segment-routing]). The SR Algorithm (refer to
Section 3.1.1 of [I-D.ietf-spring-segment-routing]) 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 [I-D.ietf-spring-segment-routing]) or BGP EPE Peering SID (as defined in [I-D.ietf-idr-bgppls-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 [I-D.ietf-spring-segment-routing]) or BGP EPE Peering SID (as defined in [I-D.ietf-idr-bgppls-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 [I-D.ietf-spring-segment-routing]) or BGP EPE Peering SID (as defined in [I-D.ietf-idr-bgppls-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 [I-D.ietf-spring-segment-routing]) 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 [I-D.ietf-spring-segment-routing]) 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 first segment the traffic will be directed toward along the SR explicit path.
- The last Segment represents the bottommost label or the last IPv6 segment the traffic will be directed toward along the SR explicit 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 SID-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 SID-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 SID-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) later in this document for more details.
5. Validity of a Candidate Path

5.1. Explicit Candidate Path

An explicit candidate path is associated with a SID-List or a set of SID-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 SID list is external to the SR Policy headend. The SR Policy headend does not compute the SID list. The SR Policy headend only confirms its validity.

A SID-List of an explicit candidate path MUST be declared invalid when:

- It is empty.
- Its weight is 0.
- The headend is unable to resolve the first SID into one or more outgoing interface(s) and next-hop(s).
- The headend is unable to resolve 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 resolve" 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 resolve 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 A or B 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 SID-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 SID-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 SID-List ("solution SID-List") that solves this optimization problem.

The headend re-computes the solution SID-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 SID list.

6. Binding SID

The Binding SID (BSID) is fundamental to Segment Routing [I-D.ietf-spring-segment-routing]. It provides scaling, network opacity and service independence. [I-D.filsfils-spring-sr-policy-considerations] illustrates some of these benefits.
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 [I-D.ietf-spring-segment-routing].

When the specified BSID is not available (optionally is not in the SRLB), an alert message is 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, 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 cannot be used as an identification of an SR Policy.

6.2.1. Frequent use-cases : 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 is triggered. Other candidate paths can 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 SID-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 and [I-D.ali-spring-sr-traffic-accounting] covers these aspects in the broader context of traffic accounting in a SR network.

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.

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 SID-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 in 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 (endpoint = N, color = C) of SID-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 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 SID-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 (endpoint = N, color = C1) of SID list <S1, S2, S3> and BSID B1.
- has a valid SR Policy P2 to (endpoint = N, color = C2) of SID 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 (endpoint N, color C).

In this section, independently to the a-priori existence of any explicit candidate path of the SR policy (N, C), 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 (N, C) 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 (endpoint N, color Ci) is triggered for each color Ci.

8.6. Per-Flow Steering

Let us assume that headend H:

- has a valid SR Policy P1 to (endpoint = N, color = C1) of SID-List <S1, S2, S3> and BSID B1.
- has a valid SR Policy P2 to (endpoint = N, color = C2) of SID-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 entry 0, 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.

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 (endpoint = N, color = C) of SID-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 10.2
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

```
1----2-----6----7
 |    |     |    |
4----3-----9----8
```

Figure 1: Local protection using SR Policy

An SR Policy can be instantiated at node 2 to protect the link 2to6. A typical explicit SID 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.

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). Such a backup candidate path is generally disjoint from 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 and Dhanendra Jain for their valuable comments and suggestions.

13. References

13.1. Normative References

[I-D.ietf-spring-segment-routing]


13.2. Informative References

[I-D.ali-spring-sr-traffic-accounting]

[I-D.anand-spring-poi-sr]

Filsfils, et al. Expires November 22, 2018
[I-D.bashandy-rtgwg-segment-routing-ti-lfa]
Bashandy, A., Filsfils, C., Decraene, B., Litkowski, S.,
Francois, P., and d. daniel.voyer@bell.ca, "Topology
Independent Fast Reroute using Segment Routing", draft-
bashandy-rtgwg-segment-routing-ti-lfa-04 (work in
progress), April 2018.

[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-idr-bgp-ls-segment-routing-msd]
Tantsura, J., Chunduri, U., Mirsky, G., and S. Sivabalan,
"Signaling Maximum SID Depth using Border Gateway Protocol
Link-State", draft-ietf-idr-bgp-ls-segment-routing-msd-01
(work in progress), October 2017.

[I-D.ietf-idr-bgpls-segment-routing-epe]
Previdi, S., 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-15 (work in progress), March 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-idr-te-lsp-distribution]
Previdi, S., 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-08 (work in progress), December 2017.

[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-12 (work in progress), May
2018.
[I-D.ietf-lsr-flex-algo]

[I-D.ietf-ospf-segment-routing-msd]

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

[I-D.sivabalan-pce-binding-label-sid]


Authors' Addresses
Clarence Filsfils
Cisco Systems, Inc.
Pegasus Parc
De kleetlaan 6a, DIEGEM BRABANT 1831
BELGIUM
Email: cfilsfil@cisco.com

Siva Sivabalan
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, Ontario K2K 3E8
Canada
Email: msiva@cisco.com

Shraddha Hegde
Juniper Networks, Inc.
Embassy Business Park
Bangalore, KA  560093
India
Email: shraddha@juniper.net

Daniel Voyer
Bell Canada.
671 de la gauchetiere W
Montreal, Quebec H3B 2M8
Canada
Email: daniel.voyer@bell.ca

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

Alex Bogdanov
Google, Inc.
Email: bogdanov@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

Paul Mattes  
Microsoft  
One Microsoft Way  
Redmond, WA  98052-6399  
USA  
Email: pamattes@microsoft.com

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

Segment Routing (SR) leverages the source routing paradigm. A node steers a packet through a controlled set of instructions, called segments, by prepending the packet with an SR header. In the MPLS dataplane, the SR header is instantiated through a label stack. This document specifies the forwarding behavior to allow instantiating SR over the MPLS dataplane.

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 http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on August 23, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction...................................................3
2. MPLS Instantiation of Segment Routing..........................3
   2.1. Supporting Multiple Forwarding Behaviors for the Same Prefix ...............................................................4
   2.2. SID Representation in the MPLS Forwarding Plane.........4
   2.3. Segment Routing Global Block and Local Block..............5
   2.4. Mapping a SID Index to an MPLS label......................6
   2.5. Incoming Label Collision..................................6
       2.5.1. Tie-breaking Rules...................................8
       2.5.2. Redistribution between Routing Protocol Instances...11
       2.5.2.1. Illustration...................................11
   2.6. Outgoing Label Collision-------------------------------11
   2.7. PUSH, CONTINUE, and NEXT.................................12
       2.7.1. PUSH................................................12
       2.7.2. CONTINUE............................................12
       2.7.3. NEXT................................................12
   2.8. MPLS Label downloaded to FIB corresponding to Global and Local SIDs....................................................13
   2.9. Active Segment...........................................13
   2.10. Forwarding behavior for Global SIDs.....................13
       2.10.1. Forwarding Behavior for PUSH and CONTINUE Operation for Global SIDs.....................................................13
       2.10.2. Forwarding Behavior for NEXT Operation for Global SIDs................................................................15
The Segment Routing architecture [I-D.ietf-spring-segment-routing] can be directly applied to the MPLS architecture with no change in the MPLS forwarding plane. This document specifies the forwarding plane behavior to allow Segment Routing to operate on top of the MPLS data plane. This document does not address the control plane behavior. Control plane behavior is specified in other documents such as [I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions], and [I-D.ietf-ospf-ospfv3-segment-routing-extensions].

The Segment Routing problem statement is described in [RFC7855].

Co-existence of SR over MPLS forwarding plane with LDP [RFC5036] is specified in [I-D.ietf-spring-segment-routing-ldp-interop].

Policy routing and traffic engineering using segment routing can be found in [I-D. filsfils-spring-segment-routing-policy]

2. MPLS Instantiation of Segment Routing

MPLS instantiation of Segment Routing fits in the MPLS architecture as defined in [RFC3031] both from a control plane and forwarding plane perspective:
From a control plane perspective, [RFC3031] does not mandate a single signaling protocol. Segment Routing makes use of various control plane protocols such as link State IGPs [I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions] and [I-D.ietf-ospf-ospfv3-segment-routing-extensions]. The flooding mechanisms of link state IGPs fits very well with label stacking on ingress. Future control layer protocol and/or policy/configuration can be used to specify the label stack.

From a forwarding plane perspective, Segment Routing does not require any change to the forwarding plane because Segment ID (SIDs) are instantiated as MPLS labels and the Segment routing header instantiated as a stack of MPLS labels.

We call "MPLS Control Plane Client (MCC)" any control plane entity installing forwarding entries in the MPLS data plane. IGPs with SR extensions [I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions], [I-D.ietf-ospf-ospfv3-segment-routing-extensions] and LDP [RFC5036] are examples of MCCs. Local configuration and policies applied on a router are also examples of MCCs.

2.1. Supporting Multiple Forwarding Behaviors for the Same Prefix

The SR architecture does not prohibit having more than one SID for the same prefix. In fact, by allowing multiple SIDs for the same prefix, it is possible to have different forwarding behaviors (such as different paths, different ECMP/UCMP behaviors,...,etc) for the same destination.

Instantiating Segment routing over the MPLS forwarding plane fits seamlessly with this principle. An operator may assign multiple MPLS labels or indices to the same prefix and assign different forwarding behaviors to each label/SID. The MCC in the network downloads different MPLS labels/SIDs to the FIB for different forwarding behaviors. The MCC at the entry of an SR domain or at any point in the domain can choose to apply a particular forwarding behavior to a particular packet by applying the PUSH action to that packet using the corresponding SID.

2.2. SID Representation in the MPLS Forwarding Plane

When instantiating SR over the MPLS forwarding plane, a SID is represented by an MPLS label or an index [I-D.ietf-spring-segment-routing].
A global segment MUST be a label, or an index which may be mapped to an MPLS label using the SRGB of the node installing the global segment in its FIB/receiving the labeled packet. Section 2.4 specifies the procedure to map a global segment represented by an index to an MPLS label within the SRGB.

The MCC MUST ensure that any label value corresponding to any SID it installs in the forwarding plane follows the following rules:

- The label value MUST be unique within the router on which the MCC is running, i.e. the label MUST only be used to represent the SID.
- The label value MUST NOT be identical to or within the range of any reserved label value or range [reserved-MPLS], respectively.

2.3. Segment Routing Global Block and Local Block

The concepts of Segment Routing Global Block (SRGB) and global SID are explained in [I-D.ietf-spring-segment-routing]. In general, the SRGB need not be a contiguous range of labels.

For the rest of this document, the SRGB is specified by the list of MPLS Label ranges \([Ll(1),Lh(1)], [Ll(2),Lh(2)], \ldots, [Ll(k),Lh(k)]\) where \(Ll(i) =< Lh(i)\).

The following rules apply to the list of MPLS ranges representing the SRGB:

- The list of ranges comprising the SRGB MUST NOT overlap.
- Every range in the list of ranges specifying the SRGB MUST NOT cover or overlap with a reserved label value or range [reserved-MPLS], respectively.
- If the SRGB of a node does not conform to the structure specified in this section or to the previous two rules, then this SRGB is completely ignored and the node is treated as if it does not have an SRGB.
- The list of label ranges MUST only be used to instantiate global SIDs into the MPLS forwarding plane.

Local segments MAY be allocated from the Segment Routing Local Block (SRLB) [I-D.ietf-spring-segment-routing] or from any unused label as long as it does not use a reserved label. The SRLB consists of the range of local labels reserved by the node for certain local segments. In a controller-driven network, some controllers or...
applications MAY use the control plane to discover the available set of local SIDs on a particular router [I.D. filsfils-spring-segment-routing-policy]. Just like SRGB, the SRLB need not be a single contiguous range of label, except the SRGB MUST only be used to instantiate global SIDs into the MPLS forwarding plane. Hence it is specified the same way and follows the same rules SRGB is specified above in this sub-section.

2.4. Mapping a SID Index to an MPLS label

This sub-section specifies how the MPLS label value is calculated given the index of a SID. The value of the index is determined by an MCC such as IS-IS [I-D.ietf-isis-segment-routing-extensions] or OSPF [I-D.ietf-ospf-segment-routing-extensions]. This section only specifies how to map the index to an MPLS label. The calculated MPLS label is downloaded to the FIB, sent out with a forwarded packet, or both.

Consider a SID represented by the index "I". Consider an SRGB as specified in Section 2.3. The total size of the SRGB, represented by the variable "Size", is calculated according to the formula:

\[
\text{size} = \text{Lh}(1) - \text{Ll}(1) + 1 + \text{Lh}(2) - \text{Ll}(2) + 1 + \ldots + \text{Lh}(k) - \text{Ll}(k) + 1
\]

The following rules MUST be applied by the MCC when calculating the MPLS label value corresponding the SID index value "I".

- 0 =< I < size. If the index "I" does not satisfy the previous inequality, then the label cannot be calculated.

- The label value corresponding to the SID index "I" is calculated as follows

  - j = 1, temp = 0
  - While temp + \text{Lh}(j) - \text{Ll}(j) < I
    - temp = temp + \text{Lh}(j) - \text{Ll}(j) + 1
    - j = j+1
  - label = I - temp + \text{Ll}(j)

2.5. Incoming Label Collision

MPLS Architecture [RFC3031] defines Forwarding Equivalence Class (FEC) as the set of packets which are forwarded in the same manner...
Internet-Draft        Segment Routing with MPLS           February 2018

(e.g., over the same path, with the same forwarding treatment) and are bound to the same MPLS incoming (local) label. In Segment-Routing MPLS, local label serves as the SID (possibly via an index indirection) for given FEC. 

We define Segment Routing (SR) FEC as one of the following [I-D.ietf-spring-segment-routing]:

- (Prefix, Routing Instance, Topology, Algorithm), where a topology is identified by a set of links with metrics. For the purpose of incoming label collision resolution, the same numerical value SHOULD be used on all routers to identify the same set of links with metrics. For MCCs where the "Topology" and/or "Algorithm" fields are not defined, the numerical value of zero MUST be used for these two fields. For the purpose of incoming label collision resolution, a routing instance is identified by a single incoming label downloader to FIB. Two MCCs running on the same router are considered different routing instances if the only way the two instances can know about the other’s incoming labels is through redistribution. The numerical value used to identify a routing instance MAY be derived from other configuration or MAY be explicitly configured. If it is derived from other configuration, then the same numerical value SHOULD be derived from the same configuration as long as the configuration survives router reload. If the derived numerical value varies for the same configuration, then an implementation SHOULD make numerical value used to identify a routing instance configurable.

- (next-hop, outgoing interface), where the outgoing interface is physical or virtual.

- (Endpoint, Color) representing an SR policy [I.D. filsfils-spring-segment-routing-policy]

This section covers handling the scenario where, because of an error/misconfiguration, more than one SR FEC as defined in this section, map to the same incoming MPLS label.

An incoming label collision occurs if the SIDs of the set of FECs (FEC1, FEC2,..., FECk) maps to the same incoming SR MPLS label "L1".

The objective of the following steps is to deterministically install in the MPLS Incoming Label MAP, also known as label FIB, a single FEC with the incoming label "L1". Remaining FECs may be installed in the IP FIB without incoming label.
The procedure in this section relies completely on the local FEC and label database within a given router.

The collision resolution procedure is as follows

1. Given the SIDs of the set of FECs, \{FEC1, FEC2, ..., FECk\} map to the same MPLS label "L1".

2. Within an MCC, apply tie-breaking rules to select one FEC only and assign the label to it. The losing FECs are handled as if no labels are attached to them. The losing FECs with a non-zero algo are not installed in FIB.
   a. If the same set of FECs are attached to the same label "L1", then the tie-breaking rules MUST always select the same FEC irrespective of the order in which the FECs and the label "L1" are received. In other words, the tie-breaking rule MUST be deterministic. For example, a first-come-first-serve tie-breaking is not allowed.

3. If there is still collision between the FECs belonging to different MCCs, then re-apply the tie-breaking rules to the remaining FECs to select one FEC only and assign the label to that FEC.

4. Install into the IP FIB the selected FEC and its incoming label in the label FIB.

5. The remaining FECs with a zero algorithm are installed in the FIB natively, such as pure IP entries in case of Prefix FEC, without any incoming labels corresponding to their SIDs. The remaining FECs with a non-zero algorithm are not installed in the FIB.

2.5.1. Tie-breaking Rules

The default tie-breaking rules SHOULD be as follows:

1. if FEC1 has the lowest FEC administrative distance among the competing FEC’s as defined in this section below, filter away all the competing FEC’s with higher administrative distance.

2. if more than one competing FEC remains after step 1, sort them and select the smallest numerical FEC value.

These rules deterministically select the FEC to install in the MPLS forwarding plane for the given incoming label.
This document defines the default tie breaking rules that SHOULD be implemented. An implementation may choose to implement additional tie-breaking rules. All routers in a routing domain SHOULD use the same tie-breaking rules to maximize forwarding consistency.

Each FEC is assigned an administrative distance. The FEC administrative distance is encoded as an 8-bit value. The lower the value, the better the administrative distance.

The default FEC administrative distance order starting from the lowest value SHOULD be:

- Explicit SID assignment to a FEC that maps to a label outside the SRGB irrespective of the owner MCC. An explicit SID assignment is a static assignment of a label to a FEC such that the assignment survives router reboot.
  - An example of explicit SID allocation is static assignment of a specific label to an adjacency SID.
  - An implementation of explicit SID assignment MUST guarantee collision freeness on the same router.

- Dynamic SID assignment:
  - For all FEC types except for SR policy, use the default administrative distance depending on the implementation.
  - Binding SID [I-D.ietf-spring-segment-routing] assigned to SR Policy.

A user SHOULD ensure that the same administrative distance preference is used on all routers to maximize forwarding consistency.

The numerical sort across FEC’s SHOULD be performed as follows:

- Each FEC is assigned a FEC type encoded in 8 bits. The following are the type code point for each SR FEC defined at the beginning of this Section:
  - 120: (Prefix, Routing Instance, Topology, Algorithm)
  - 130: (next-hop, outgoing interface)
  - 140: (Endpoint, Color) representing an SR policy

- The fields of each FEC are encoded as follows.
Routing Instance ID represented by 16 bits. For routing instances that are identified by less than 16 bits, encode the Instance ID in the least significant bits while the most significant bits are set to zero.

Address Family represented by 8 bits, where IPv4 encoded as 100 and IPv6 is encoded as 110.

All addresses are represented in 128 bits as follows:
- IPv6 address is encoded natively
- IPv4 address is encoded in the most significant bits and the remaining bits are set to zero.

All prefixes are represented by 128.
- A prefix is encoded in the most significant bits and the remaining bits are set to zero.
- The prefix length is encoded before the prefix.

Topology ID is represented by 16 bits. For routing instances that identify topologies using less than 16 bits, encode the topology ID in the least significant bits while the most significant bits are set to zero.

Algorithm is encoded in a 16 bits field.

The Color ID is encoded using 16 bits.

Choose the set of FECs of the smallest FEC type code point.

Out of these FECs, choose the FECs with the smallest address family code point.

Encode the remaining set of FECs as follows:
- Prefix, Routing Instance, Topology, Algorithm: (Prefix Length, Prefix, SR Algorithm, routing_instance_id, Topology)
- (next-hop, outgoing interface): (next-hop, outgoing_interface_id)
- (Endpoint, Color): (Endpoint_address, Color_id)

Select the FEC with the smallest numerical value.
2.5.2. Redistribution between Routing Protocol Instances

The following rule SHOULD be applied when redistributing SIDs with prefixes between routing protocol instances:

- If the receiving instance’s SRGB is the same as the SRGB of origin instance, THEN
  - the index is redistributed with the route
- Else
  - the index is not redistributed and if needed it is the duty of the receiving instance to allocate a fresh index relative to its own SRGB

It is outside the scope of this document to define local node behaviors that would allow to map the original index into a new index in the receiving instance via the addition of an offset or other policy means.

2.5.2.1. Illustration

A----IS-IS----B---OSPF----C-1.1.1.1/32 (20001)

Consider the simple topology above.

- A and B are in the IS-IS domain with SRGB [16000-17000]
- B and C are in OSPF domain with SRGB [20000-21000]
- B redistributes 1.1.1.1/32 into IS-IS domain
- In that case A learns 1.1.1.1/32 as an IP leaf connected to B as usual for IP prefix redistribution
- However, according to the redistribution rule above rule, B does not advertise any index with 1.1.1.1/32 into IS-IS because the SRGB is not the same.

2.6. Outgoing Label Collision

For the determination of the outgoing label to use, the ingress node pushing new segments, and hence a stack of MPLS labels, MUST use, for
a given FEC, the same label that has been selected by the node receiving the packet with that label exposed as top label. So in case of incoming label collision on this receiving node, the ingress node MUST resolve this collision using this same "Incoming Label Collision resolution procedure", using the data of the receiving node.

In the general case, the ingress node may not have exactly have the same data of the receiving node, so the result may be different. This is under the responsibility of the network operator. But in typical case, e.g. where a centralized node or a distributed link state IGP is used, all nodes would have the same database. However to minimize the chance of misforwarding, a FEC that loses its incoming label to the tie-breaking rules specified in Section 2.5 MUST NOT be installed in FIB with an outgoing segment routing label based on the SID corresponding to the lost incoming label.

2.7. PUSH, CONTINUE, and NEXT

PUSH, NEXT, and CONTINUE are operations applied by the forwarding plan. The specifications of these operations can be found in [I-D.ietf-spring-segment-routing]. This sub-section specifies how to implement each of these operations in the MPLS forwarding plane.

2.7.1. PUSH

PUSH corresponds to pushing one or more labels on top of an incoming packet then sending it out of a particular physical interface or virtual interface, such as UDP tunnel [RFC7510] or L2TPv3 tunnel [RFC4817], towards a particular next-hop. Sections 2.10 and 2.11 specify additional details about forwarding behavior.

2.7.2. CONTINUE

In the MPLS forwarding plane, the CONTINUE operation corresponds to swapping the incoming label with an outgoing label. The value of the outgoing label is calculated as specified in Sections 2.10 and 2.11.

2.7.3. NEXT

In the MPLS forwarding plane, NEXT corresponds to popping the topmost label. The action before and/or after the popping depends on the instruction associated with the active SID on the received packet prior to the popping. For example suppose the active SID in the received packet was an Adj-SID [I-D.ietf-spring-segment-routing], then on receiving the packet, the node applies NEXT operation, which corresponds to popping the top most label, and then sends the packet out of the physical or virtual interface (e.g. UDP tunnel [RFC7510])
2.8. MPLS Label downloaded to FIB corresponding to Global and Local SIDs

The label corresponding to the global SID "Si" represented by the global index "I" downloaded to FIB is used to match packets whose active segment (and hence topmost label) is "Si". The value of this label is calculated as specified in Section 2.4.

For Local SIDs, the MCC is responsible for downloading the correct label value to FIB. For example, an IGP with SR extensions I-D.ietf-isis-segment-routing-extensions, I-D.ietf-ospf-segment-routing-extensions] allocates and downloads the MPLS label corresponding to an IGP-adjacency-SID [I-D.ietf-spring-segment-routing].

2.9. Active Segment

When instantiated in the MPLS domain, the active segment on a packet corresponds to the topmost label on the packet that is calculated according to the procedure specified in Sections 2.10 and 2.11. When arriving at a node, the topmost label corresponding to the active SID matches the MPLS label downloaded to FIB as specified in Section 2.8.

2.10. Forwarding behavior for Global SIDs

This section specifies forwarding behavior, including the outgoing label(s) calculations corresponding to a global SID when applying PUSH, CONTINUE, and NEXT operations in the MPLS forwarding plane.

This document covers the calculation of outgoing label for the top label only. The case where outgoing label is not the top label and is part of a stack of labels that instantiates a routing policy or a traffic engineering tunnel is covered in other documents such as [I.D.filsfils-spring-segment-routing-policy].

2.10.1. Forwarding Behavior for PUSH and CONTINUE Operation for Global SIDs

Suppose an MCC on a router "R0" determines that PUSH or CONTINUE operation is to be applied to an incoming packet whose active SID is the global SID "Si" represented by the global index "I" and owned by the router Ri before sending the packet towards a neighbor "N" directly connected to "R0" through a physical or virtual interface such as UDP tunnel [RFC7510] or L2TPv3 tunnel [RFC4817].
The method by which the MCC on router "R0" determines that PUSH or CONTINUE operation must be applied using the SID "Si" is beyond the scope of this document. An example of a method to determine the SID "Si" for PUSH operation is the case where IS-IS [I-D.ietf-isis-segment-routing-extensions] receives the prefix-SID "Si" sub-TLV advertised with prefix "P/m" in TLV 135 and the destination address of the incoming IPv4 packet is covered by the prefix "P/m".

For CONTINUE operation, an example of a method to determine the SID "Si" is the case where IS-IS [I-D.ietf-isis-segment-routing-extensions] receives the prefix-SID "Si" sub-TLV advertised with prefix "P" in TLV 135 and the top label of the incoming packet matches the MPLS label in FIB corresponding to the SID "Si" on the router "R0".

The forwarding behavior for PUSH and CONTINUE corresponding to the SID "Si"

- If the neighbor "N" does not support SR or "I" does not satisfy the inequality specified in Section 2.4 for the SRGB of the neighbor "N"
  - If it is possible to send the packet towards the neighbor "N" using standard MPLS forwarding behavior as specified in (RFC3031) and [RFC3032], then forward the packet. The method by which a router decides whether it is possible to send the packet to "N" or not is beyond the scope of this document. For example, the router "R0" can use the downstream label determined by another MCC, such as LDP [RFC5036], to send the packet.
  - Else if there are other useable next-hops, then use other next-hops to forward the incoming packet. The method by which the router "R0" decides on the possibility of using other next-hops is beyond the scope of this document. For example, the MCC on "R0" may chose the send an IPv4 packet without pushing any label to another next-hop.
  - Otherwise drop the packet.
- Else
  - Calculate the outgoing label as specified in Section 2.4 using the SRGB of the neighbor "N"
  - If the operation is PUSH
. Push the calculated label according the MPLS label pushing rules specified in [RFC3032]

   o Else

   . swap the incoming label with the calculated label according to the label swapping rules in [RFC3032]

   o Send the packet towards the neighbor "N"

2.10.2. Forwarding Behavior for NEXT Operation for Global SIDs

As specified in Section 2.7.3 NEXT operation corresponds to popping the top most label. The forwarding behavior is as follows

   o Pop the topmost label

   o Apply the instruction associated with the incoming label prior to popping

The action on the packet after popping the topmost label depends on the instruction associated with the incoming label as well as the contents of the packet right underneath the top label that got popped. Examples of NEXT operation are described in Section 3.

2.11. Forwarding Behavior for Local SIDs

This section specifies the forwarding behavior for local SIDs when SR is instantiated over the MPLS forwarding plane.

2.11.1. Forwarding Behavior Corresponding to PUSH Operation on Local SIDs

Suppose an MCC on a router "R0" determines that PUSH operation is to be applied to an incoming packet using the local SID "Si" before sending the packet towards a neighbor "N" directly connected to R0 through a physical or virtual interface such as UDP tunnel [RFC7510] or L2TPv3 tunnel [RFC4817].

An example of such local SID is an IGP-Adj-SID allocated and advertised by IS-IS [I-D.ietf-isis-segment-routing-extensions]. The method by which the MCC on "R0" determines that PUSH operation is to be applied to the incoming packet is beyond the scope of this document. An example of such method is backup path used to protect
against a failure using Ti-LFA [I-D.bashandy-rtgwg-segment-routing-ti-lfa].

As mentioned in [I-D.ietf-spring-segment-routing], a local SID is specified by an MPLS label. Hence the PUSH operation for a local SID is identical to label push operation [RFC3032] using any MPLS label. The forwarding action after pushing the MPLS label corresponding to the local SID is also determined by the MCC. For example, if the PUSH operation was done to forward a packet over a backup path calculated using Ti-LFA, then the forwarding action may be sending the packet to a certain neighbor that will in turn continue to forward the packet along the backup path.

2.11.2. Forwarding Behavior Corresponding to CONTINUE Operation for Local SIDs

A local SID on a router "R0" corresponds to a local label such as an IGP adj-SID. In such scenario, the outgoing label towards a next-hop "N" is determined by the MCC running on the router "R0" and the forwarding behavior for CONTINUE operation is identical to swap operation [RFC3032] on an MPLS label.

2.11.3. Outgoing label for NEXT Operation for Local SIDs

NEXT operation for Local SIDs is identical to NEXT operation for global SIDs specified in Section 2.10.2.

3. IGP Segments Examples

Consider the network diagram of Figure 1 and the IP address and IGP Segment allocation of Figure 2. Assume that the network is running IS-IS with SR extensions [I-D.ietf-isis-segment-routing-extensions]. The following examples can be constructed.

```
+--------+
|        |
R0------R1-----R2-------R3-----R8
\         \      /       |
 R4------R5-----+      |
```

Figure 1: IGP Segments - Illustration
### 3.1. Example 1

Suppose R1 wants to send an IPv4 packet P1 to R8. In this case, R1 needs to apply PUSH operation to the IPv4 packet.

Remember that the SID index "8" is a global IGP segment attached to the IP prefix 192.0.2.8/32. Its semantic is global within the IGP domain: any router forwards a packet received with active segment 8 to the next-hop along the ECMP-aware shortest-path to the related prefix.

R2 is the next-hop along the shortest path towards R8. By applying the steps in Section 2.8 the local label downloaded to R1’s FIB corresponding to the global SID index 8 is 1008 because the SRGB of R2 is [1000,5000] as shown in Figure 2.

---

**Figure 2: IGP Address and Segment Allocation - Illustration**

**IP address allocated by the operator:**
- 192.0.2.1/32 as a loopback of R1
- 192.0.2.2/32 as a loopback of R2
- 192.0.2.3/32 as a loopback of R3
- 192.0.2.4/32 as a loopback of R4
- 192.0.2.5/32 as a loopback of R5
- 192.0.2.8/32 as a loopback of R8
- 198.51.100.9/32 as an anycast loopback of R4
- 198.51.100.9/32 as an anycast loopback of R5

**SRGB defined by the operator as 1000-5000**

**Global IGP SID indices allocated by the operator:**
- 1 allocated to 192.0.2.1/32
- 2 allocated to 192.0.2.2/32
- 3 allocated to 192.0.2.3/32
- 4 allocated to 192.0.2.4/32
- 8 allocated to 192.0.2.8/32
- 1009 allocated to 198.51.100.9/32

**Local IGP SID allocated dynamically by R2**
- for its "north" adjacency to R3: 9001
- for its "south" adjacency to R3: 9002
- for its "north" adjacency to R3: 9003
- for its "south" adjacency to R3: 9003
Because the packet is IPv4, R1 applies the PUSH operation using the label value 1008 as specified in 2.10.1. The resulting MPLS header will have the "S" bit [RFC3032] set because it is followed directly by an IPv4 packet.

The packet arrives at router R2. Because the top label 1008 corresponds to the IGP SID "8", which is the prefix-SID attached to the prefix 192.0.2.8/32 owned by the R8, then the instruction associated with the SID is "forward the packet using all ECMP/UCMP interfaces and all ECMP/UCMP next-hop(s) along the shortest path towards R8". Because R2 is not the penultimate hop, R2 applies the CONTINUE operation to the packet and sends it to R3 using one of the two links connected to R3 with top label 1008 as specified in Section 2.10.1.

R3 receives the packet with top label 1008. Because the top label 1008 corresponds to the IGP SID "8", which is the prefix-SID attached to the prefix 192.0.2.8/32 owned by the R8, then the instruction associated with the SID is "send the packet using all ECMP interfaces and all next-hop(s) along the shortest path towards R8". Because R3 is the penultimate hop, R3 applies NEXT operation then sends the packet to R8. The NEXT operation results in popping the outer label and sending the packet as a pure IPv4 packet to R8. The

In conclusion, the path followed by P1 is R1-R2--R3-R8. The ECMP-awareness ensures that the traffic be load-shared between any ECMP path, in this case the two north and south links between R2 and R3.

3.2. Example 2

Suppose the right most router R0 wants to send a packet P2 to R8 over the path <R2, (north link between R2 and R3)>. In that case, the router R0 needs to use the SID list <2, 9001, 8>. Using the calculation techniques specified in Section 2.10 and 2.11 the resulting label stack starting from the topmost label is <1002, 9001, 1008>.

The MPLS label 1002 is the MPLS instantiation of the global IGP segment index 2 attached to the IP prefix 192.0.2.2/32. Its semantic is global within the IGP domain: any router forwards a packet received with active segment 1002 to the next-hop along the shortest-path to the related prefix.

The MPLS label 9001 is a local IGP segment attached by node R2 to its north link to R3. Its semantic is local to node R2: R2 applies NEXT operation, which corresponding to popping the outer label, then
switches a packet received with active segment 9001 towards the north link to R3.

In conclusion, the path followed by P2 is R0-R1-R2-north-link-R3-R8.

3.3. Example 3

R0 may send a packet P3 along the same exact path as P2 using a different segment list <2,9003,8> which corresponds to the label stack <1002, 9003, 1008>.

9003 is a local IGP segment attached by node R2 to both its north and south links to R3. Its semantic is local to node R2: R2 applies NEXT operation, which corresponds to popping the top label, then switches a packet received with active segment 9003 towards either the north or south links to R3 (e.g. per-flow loadbalancing decision).

In conclusion, the path followed by P3 is R0-R1-R2-any-link-R3-R8.

3.4. Example 4

R0 may send a packet P4 to R8 while avoiding the links between R2 and R3 by pushing the SID list <4,8>, which corresponds to the label stack <1004, 1008>.

1004 is a global IGP segment attached to the IP prefix 192.0.2.4/32. Its semantic is global within the IGP domain: any router forwards a packet received with active segment 1004 to the next-hop along the shortest-path to the related prefix.

In conclusion, the path followed by P4 is R0-R1-R2-R4-R3-R8.

3.5. Example 5

R0 may send a packet P5 to R8 while avoiding the links between R2 and R3 and still benefiting from all the remaining shortest paths (via R4 and R5) by pushing the SID list <1009, 8> which corresponds to the label stack <2009, 1008> using the steps specified in Sections 2.10 and 2.11.

1009 is a global anycast-SID [I-D.ietf-spring-segment-routing] attached to the anycast IP prefix 198.51.100.9/32. Its semantic is global within the IGP domain: any router forwards a packet received with top label 2009 (corresponding to the active segment 1009) to the next-hop along the shortest-path to the related prefix.
In conclusion, the path followed by P5 is either R0-R1-R2-R4-R3-R8 or R0-R1-R2-R5-R3-R8.

4. IANA Considerations

This document does not make any request to IANA.

5. Manageability Considerations

This document describes the applicability of Segment Routing over the MPLS data plane. Segment Routing does not introduce any change in the MPLS data plane. Manageability considerations described in [I-D.ietf-spring-segment-routing] applies to the MPLS data plane when used with Segment Routing.

6. Security Considerations

This document does not introduce additional security requirements and mechanisms other than the ones described in [I-D.ietf-spring-segment-routing].

7. Contributors

The following contributors have substantially helped the definition and editing of the content of this document:
8. Acknowledgements

The authors would like to thank Les Ginsberg and Shah Himanshu for their comments on this document.

This document was prepared using 2-Word-v2.0.template.dot.

9. References

9.1. Normative References


9.2. Informative References


Filsfils, et al. Expires August 23, 2018
Internet-Draft  Segment Routing with MPLS  February 2018

[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>.

[RFC4817] Townsley, M., Pignataro, C., Wainner, S., Seely, T., Young,
T., "Encapsulation of MPLS over Layer 2 Tunneling Protocol
<https://www.rfc-editor.org/info/rfc4817>

[I-D.ietf-idr-segment-routing-te-policy] Previdi, S., Filsfils, C.,
Mattes, P., Rosen, E., Lin, S., " Advertising Segment
Routing Policies in BGP", draft- ietf-idr-segment-routing-
te-policy-00 (work in progress), July 2017

[I.D. filsfs-spring-segment-routing-policy] Filsfils, C.,
Sivabalan, S., Raza, K., Liste, J., Clad, F., Voyer, D.,
Lin, S., Bogdanov, A., Horneffer, M., Steinberg, D.,
Decraene, B., Litkowski, S., " Segment Routing Policy for
Traffic Engineering", draft-filsfils-spring-segment-
routing-policy-01 (work in progress), July 2017
Authors’ Addresses

Ahmed Bashandy
Cisco Systems, Inc.
170, West Tasman Drive
San Jose, CA  95134
US
Email: bashandy@cisco.com

Clarence Filsfils (editor)
Cisco Systems, Inc.
Brussels
BE
Email: cfilsfil@cisco.com

Stefano Previdi (editor)
Cisco Systems, Inc.
Italy
Email: stefano@previdi.net

Bruno Decraene
Orange
FR
Email: bruno.de craene@orange.com

Stephane Litkowski
Orange
FR
Email: stephane.litkowski@orange.com

Rob Shakir
Google
US
Email: robjs@google.com
Segment Routing with MPLS data plane
draft-ietf-spring-segment-routing-mpls-14

Abstract

Segment Routing (SR) leverages the source routing paradigm. A node steers a packet through a controlled set of instructions, called segments, by prepending the packet with an SR header. In the MPLS dataplane, the SR header is instantiated through a label stack. This document specifies the forwarding behavior to allow instantiating SR over the MPLS dataplane.

Status of this Memo

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

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

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

This Internet-Draft will expire on December 11, 2018.
Copyright Notice

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

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

Table of Contents

1. Introduction...................................................3
   1.1. Requirements Language.....................................3
2. MPLS Instantiation of Segment Routing..........................3
   2.1. Multiple Forwarding Behaviors for the Same Prefix.........4
   2.2. SID Representation in the MPLS Forwarding Plane...........4
   2.3. Segment Routing Global Block and Local Block...............5
   2.4. Mapping a SID Index to an MPLS label......................6
   2.5. Incoming Label Collision..................................7
      2.5.1. Tie-breaking Rules...................................8
      2.5.2. Redistribution between Routing Protocol Instances...11
         2.5.2.1. Illustration...................................11
         2.5.2.2. Illustration 2.................................12
   2.6. Outgoing Label Collision..................................12
   2.7. PUSH, CONTINUE, and NEXT.................................12
      2.7.1. PUSH................................................13
      2.7.2. CONTINUE............................................13
      2.7.3. NEXT................................................13
   2.8. MPLS Label Downloaded to FIB for Global and Local SIDs...13
   2.9. Active Segment...........................................13
   2.10. Forwarding behavior for Global SIDs.....................14
      2.10.1. Forwarding for PUSH and CONTINUE of Global SIDs....14
      2.10.2. Forwarding for NEXT Operation for Global SIDs......15
   2.11. Forwarding Behavior for Local SIDs.......................16
      2.11.1. Forwarding for PUSH Operation on Local SIDs.........16
      2.11.2. Forwarding for CONTINUE Operation for Local SIDs...16
      2.11.3. Outgoing label for NEXT Operation for Local SIDs...16
3. IGP Segments Examples.........................................17
   3.1. Example 1................................................18
   3.2. Example 2................................................19
1. Introduction

The Segment Routing architecture [I-D.ietf-spring-segment-routing] can be directly applied to the MPLS architecture with no change in the MPLS forwarding plane. This document specifies the forwarding plane behavior to allow Segment Routing to operate on top of the MPLS data plane. This document does not address the control plane behavior. Control plane behavior is specified in other documents such as [I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions], and [I-D.ietf-ospf-ospfv3-segment-routing-extensions].

The Segment Routing problem statement is described in [RFC7855].

Co-existence of SR over MPLS forwarding plane with LDP [RFC5036] is specified in [I-D.ietf-spring-segment-routing-ldp-interop].

Policy routing and traffic engineering using segment routing can be found in [I.D. filsfils-spring-segment-routing-policy]

1.1. Requirements Language

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

2. MPLS Instantiation of Segment Routing

MPLS instantiation of Segment Routing fits in the MPLS architecture as defined in [RFC3031] both from a control plane and forwarding plane perspective:
o From a control plane perspective, [RFC3031] does not mandate a single signaling protocol. Segment Routing makes use of various control plane protocols such as link state IGPs [I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions] and [I-D.ietf-ospf-ospfv3-segment-routing-extensions]. The flooding mechanisms of link state IGPs fits very well with label stacking on ingress. Future control layer protocol and/or policy/configuration can be used to specify the label stack.

o From a forwarding plane perspective, Segment Routing does not require any change to the forwarding plane because Segment IDs (SIDs) are instantiated as MPLS labels and the Segment routing header instantiated as a stack of MPLS labels.

We call "MPLS Control Plane Client (MCC)" any control plane entity installing forwarding entries in the MPLS data plane. IGPs with SR extensions [I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions], [I-D.ietf-ospf-ospfv3-segment-routing-extensions] and LDP [RFC5036] are examples of MCCs. Local configuration and policies applied on a router are also examples of MCCs.

2.1. Multiple Forwarding Behaviors for the Same Prefix

The SR architecture does not prohibit having more than one SID for the same prefix. In fact, by allowing multiple SIDs for the same prefix, it is possible to have different forwarding behaviors (such as different paths, different ECMP/UCMP behaviors, ... , etc) for the same destination.

Instantiating Segment routing over the MPLS forwarding plane fits seamlessly with this principle. An operator may assign multiple MPLS labels or indices to the same prefix and assign different forwarding behaviors to each label/SID. The MCC in the network downloads different MPLS labels/SIDs to the FIB for different forwarding behaviors. The MCC at the entry of an SR domain or at any point in the domain can choose to apply a particular forwarding behavior to a particular packet by applying the PUSH action to that packet using the corresponding SID.

2.2. SID Representation in the MPLS Forwarding Plane

When instantiating SR over the MPLS forwarding plane, a SID is represented by an MPLS label or an index [I-D.ietf-spring-segment-routing].
A global segment MUST be a label, or an index which may be mapped to an MPLS label within the Segment Routing Global Block (SRGB) of the node installing the global segment in its FIB/receiving the labeled packet. Section 2.4 specifies the procedure to map a global segment represented by an index to an MPLS label within the SRGB.

The MCC MUST ensure that any label value corresponding to any SID it installs in the forwarding plane follows the following rules:

- The label value MUST be unique within the router on which the MCC is running. i.e. the label MUST only be used to represent the SID and MUST NOT be used to represent more than one SID or for any other forwarding purpose on the router.

- The label value MUST NOT come from the range of special purpose labels [RFC7274].

2.3. Segment Routing Global Block and Local Block

The concepts of Segment Routing Global Block (SRGB) and global SID are explained in [I-D.ietf-spring-segment-routing]. In general, the SRGB need not be a contiguous range of labels.

For the rest of this document, the SRGB is specified by the list of MPLS Label ranges \([Ll(1),Lh(1)], [Ll(2),Lh(2)], ..., [Ll(k),Lh(k)]\) where \(Ll(i) \leq Lh(i)\).

The following rules apply to the list of MPLS ranges representing the SRGB:

- The list of ranges comprising the SRGB MUST NOT overlap.

- Every range in the list of ranges specifying the SRGB MUST NOT cover or overlap with a reserved label value or range [RFC7274], respectively.

- If the SRGB of a node does not conform to the structure specified in this section or to the previous two rules, then this SRGB MUST be completely ignored by all routers in the routing domain and the node MUST be treated as if it does not have an SRGB.

- The list of label ranges MUST only be used to instantiate global SIDs into the MPLS forwarding plane.

Local segments MAY be allocated from the Segment Routing Local Block (SRLB) [I-D.ietf-spring-segment-routing] or from any unused label as long as it does not use a special purpose label. The SRLB consists of
the range of local labels reserved by the node for certain local segments. In a controller-driven network, some controllers or applications MAY use the control plane to discover the available set of local SIDs on a particular router [I.D. filsfils-spring-segment-routing-policy]. Just like SRGB, the SRLB need not be a single contiguous range of labels, except the SRGB MUST only be used to instantiate global SIDs into the MPLS forwarding plane. Hence it is specified the same way and follows the same rules SRGB is specified above in this sub-section.

2.4. Mapping a SID Index to an MPLS label

This sub-section specifies how the MPLS label value is calculated given the index of a SID. The value of the index is determined by an MCC such as IS-IS [I-D.ietf-isis-segment-routing-extensions] or OSPF [I-D.ietf-ospf-segment-routing-extensions]. This section only specifies how to map the index to an MPLS label. The calculated MPLS label is downloaded to the FIB, sent out with a forwarded packet, or both.

Consider a SID represented by the index "I". Consider an SRGB as specified in Section 2.3. The total size of the SRGB, represented by the variable "Size", is calculated according to the formula:

\[
\text{size} = \text{Lh}(1) - \text{Ll}(1) + 1 + \text{Lh}(2) - \text{Ll}(2) + 1 + \ldots + \text{Lh}(k) - \text{Ll}(k) + 1
\]

The following rules MUST be applied by the MCC when calculating the MPLS label value corresponding the SID index value "I".

- 0 \leq I < size. If the index "I" does not satisfy the previous inequality, then the label cannot be calculated.

- The label value corresponding to the SID index "I" is calculated as follows
  - j = 1 , temp = 0
  - While temp + \text{Lh}(j) - \text{Ll}(j) < I
    - temp = temp + \text{Lh}(j) - \text{Ll}(j) + 1
    - j = j+1
  - label = I - temp + \text{Ll}(j)
2.5. Incoming Label Collision

MPLS Architecture [RFC3031] defines Forwarding Equivalence Class (FEC) term as the set of packets with similar and/or identical characteristics which are forwarded the same way and are bound to the same MPLS incoming (local) label. In Segment-Routing MPLS, local label serves as the SID for given FEC.

We define Segment Routing (SR) FEC as one of the following [I-D.ietf-spring-segment-routing]:

- (Prefix, Routing Instance, Topology, Algorithm), where a topology is identified by a set of links with metrics. For the purpose of incoming label collision resolution, the same numerical value SHOULD be used on all routers to identify the same set of links with metrics. For MCCs where the "Topology" and/or "Algorithm" fields are not defined, the numerical value of zero MUST be used for these two fields. For the purpose of incoming label collision resolution, a routing instance is identified by a single incoming label downloader to FIB. Two MCCs running on the same router are considered different routing instances if the only way the two instances can know about the other's incoming labels is through redistribution. The numerical value used to identify a routing instance MAY be derived from other configuration or MAY be explicitly configured. If it is derived from other configuration, then the same numerical value SHOULD be derived from the same configuration as long as the configuration survives router reload. If the derived numerical value varies for the same configuration, then an implementation SHOULD make numerical value used to identify a routing instance configurable.

- (next-hop, outgoing interface), where the outgoing interface is physical or virtual.

- (Endpoint, Color) representing an SR policy [I.D. filsfils-spring-segment-routing-policy]

This section covers handling the scenario where, because of an error/misconfiguration, more than one SR FEC as defined in this section, map to the same incoming MPLS label.

An incoming label collision occurs if the SIDs of the set of FECs (FEC1, FEC2, ..., FECk) maps to the same incoming SR MPLS label "L1".

The objective of the following steps is to deterministically install in the MPLS Incoming Label MAP, also known as label FIB, a single FEC
with the incoming label "L1". Remaining FECs may be installed in the IP FIB without incoming label.

The procedure in this section relies completely on the local FEC and label database within a given router.

The collision resolution procedure is as follows:

1. Given the SIDs of the set of FECs, \{FEC1, FEC2, ..., FECk\} map to the same MPLS label "L1".

2. Within an MCC, apply tie-breaking rules to select one FEC only and assign the label to it. The losing FECs are handled as if no labels are attached to them. The losing FECs with a non-zero algorithm are not installed in FIB.
   a. If the same set of FECs are attached to the same label "L1", then the tie-breaking rules MUST always select the same FEC irrespective of the order in which the FECs and the label "L1" are received. In other words, the tie-breaking rule MUST be deterministic. For example, a first-come-first-serve tie-breaking is not allowed.

3. If there is still collision between the FECs belonging to different MCCs, then re-apply the tie-breaking rules to the remaining FECs to select one FEC only and assign the label to that FEC.

4. Install into the IP FIB the selected FEC and its incoming label in the label FIB.

5. The remaining FECs with a zero algorithm are installed in the FIB natively, such as pure IP entries in case of Prefix FEC, without any incoming labels corresponding to their SIDs. The remaining FECs with a non-zero algorithm are not installed in the FIB.

2.5.1. Tie-breaking Rules

The default tie-breaking rules SHOULD be as follows:

1. if FECi has the lowest FEC administrative distance among the competing FECs as defined in this section below, filter away all the competing FECs with higher administrative distance.

2. if more than one competing FEC remains after step 1, select the smallest numerical FEC value.
These rules deterministically select the FEC to install in the MPLS forwarding plane for the given incoming label.

This document defines the default tie breaking rules that SHOULD be implemented. An implementation MAY choose to implement additional tie-breaking rules. All routers in a routing domain SHOULD use the same tie-breaking rules to maximize forwarding consistency.

Each FEC is assigned an administrative distance. The FEC administrative distance is encoded as an 8-bit value. The lower the value, the better the administrative distance.

The default FEC administrative distance order starting from the lowest value SHOULD be

- Explicit SID assignment to a FEC that maps to a label outside the SRGB irrespective of the owner MCC. An explicit SID assignment is a static assignment of a label to a FEC such that the assignment survives router reboot.
  - An example of explicit SID allocation is static assignment of a specific label to an adj-SID.
  - An implementation of explicit SID assignment MUST guarantee collision freeness on the same router

- Dynamic SID assignment:
  - For all FEC types except for SR policy, use the default administrative distance depending on the implementation
  - Binding SID [I-D.ietf-spring-segment-routing] assigned to SR Policy

A user SHOULD ensure that the same administrative distance preference is used on all routers to maximize forwarding consistency.

The numerical sort across FECs SHOULD be performed as follows:

- Each FEC is assigned a FEC type encoded in 8 bits. The following are the type code point for each SR FEC defined at the beginning of this Section:
  - 120: (Prefix, Routing Instance, Topology, Algorithm)
  - 130: (next-hop, outgoing interface)
- 140: (Endpoint, Color) representing an SR policy
- The fields of each FEC are encoded as follows
  - Routing Instance ID represented by 16 bits. For routing instances that are identified by less than 16 bits, encode the Instance ID in the least significant bits while the most significant bits are set to zero
  - Address Family represented by 8 bits, where IPv4 encoded as 100 and IPv6 is encoded as 110
  - All addresses are represented in 128 bits as follows
    - IPv6 address is encoded natively
    - IPv4 address is encoded in the most significant bits and the remaining bits are set to zero
  - All prefixes are represented by 128.
    - A prefix is encoded in the most significant bits and the remaining bits are set to zero.
    - The prefix length is encoded before the prefix
  - Topology ID is represented by 16 bits. For routing instances that identify topologies using less than 16 bits, encode the topology ID in the least significant bits while the most significant bits are set to zero
  - Algorithm is encoded in a 16 bits field.
  - The Color ID is encoded using 16 bits
- Choose the set of FECs of the smallest FEC type code point
- Out of these FECs, choose the FECs with the smallest address family code point
- Encode the remaining set of FECs as follows
  - Prefix, Routing Instance, Topology, Algorithm: (Prefix Length, Prefix, SR Algorithm, routing_instance_id, Topology)
  - (next-hop, outgoing interface): (next-hop, outgoing_interface_id)
o (Endpoint, Color): (Endpoint_address, Color_id)

o Select the FEC with the smallest numerical value

2.5.2. Redistribution between Routing Protocol Instances

The following rule SHOULD be applied when redistributing SIDs with prefixes between routing protocol instances:

- If the receiving instance’s SRGB is the same as the SRGB of origin instance, then
  - the index is redistributed with the route
- Else
  - the index is not redistributed and if needed it is the duty of the receiving instance to allocate a fresh index relative to its own SRGB. Note that in that case, the receiving instance MUST compute its local label according section 2.4 and install it in FIB.

It is outside the scope of this document to define local node behaviors that would allow to map the original index into a new index in the receiving instance via the addition of an offset or other policy means.

2.5.2.1. Illustration

A----IS-IS----B---OSPF----C-192.0.2.1/32 (20001)

Consider the simple topology above.

- A and B are in the IS-IS domain with SRGB [16000-17000]
- B and C are in OSPF domain with SRGB [20000-21000]
- B redistributes 192.0.2.1/32 into IS-IS domain
- In that case A learns 192.0.2.1/32 as an IP leaf connected to B as usual for IP prefix redistribution
However, according to the redistribution rule above rule, B decides not to advertise any index with 192.0.2.1/32 into IS-IS because the SRGB is not the same.

### 2.5.2.2. Illustration 2

Consider the example in the illustration described in Section 2.5.2.1.

When router B redistributes the prefix 192.0.2.1/32, router B decides to allocate and advertise the same index 1 with the prefix 192.0.2.1/32 within the SRGB of the IS-IS domain, index 1 corresponds to the local label 16001.

Hence according to the redistribution rule above, router B programs the incoming label 16001 in its FIB to match traffic arriving from the IS-IS domain destined to the prefix 192.0.2.1/32.

### 2.6. Outgoing Label Collision

For the determination of the outgoing label to use, the ingress node pushing new segments, and hence a stack of MPLS labels, MUST use, for a given FEC, the same label that has been selected by the node receiving the packet with that label exposed as top label. So in case of incoming label collision on this receiving node, the ingress node MUST resolve this collision using this same "Incoming Label Collision resolution procedure", using the data of the receiving node.

In the general case, the ingress node may not have exactly have the same data of the receiving node, so the result may be different. This is under the responsibility of the network operator. But in typical case, e.g. where a centralized node or a distributed link state IGP is used, all nodes would have the same database. However to minimize the chance of misforwarding, a FEC that loses its incoming label to the tie-breaking rules specified in Section 2.5 MUST NOT be installed in FIB with an outgoing segment routing label based on the SID corresponding to the lost incoming label.

### 2.7. PUSH, CONTINUE, and NEXT

PUSH, NEXT, and CONTINUE are operations applied by the forwarding plane. The specifications of these operations can be found in [I-
D.ietf-spring-segment-routing]. This sub-section specifies how to implement each of these operations in the MPLS forwarding plane.

2.7.1. PUSH

PUSH corresponds to pushing one or more labels on top of an incoming packet then sending it out of a particular physical interface or virtual interface, such as UDP tunnel [RFC7510] or L2TPv3 tunnel [RFC4817], towards a particular next-hop. Sections 2.10 and 2.11 specify additional details about forwarding behavior.

2.7.2. CONTINUE

In the MPLS forwarding plane, the CONTINUE operation corresponds to swapping the incoming label with an outgoing label. The value of the outgoing label is calculated as specified in Sections 2.10 and 2.11.

2.7.3. NEXT

In the MPLS forwarding plane, NEXT corresponds to popping the topmost label. The action before and/or after the popping depends on the instruction associated with the active SID on the received packet prior to the popping. For example, suppose the active SID in the received packet was an Adj-SID [I-D.ietf-spring-segment-routing], then on receiving the packet, the node applies NEXT operation, which corresponds to popping the top most label, and then sends the packet out of the physical or virtual interface (e.g. UDP tunnel [RFC7510] or L2TPv3 tunnel [RFC4817])) towards the next-hop corresponding to the adj-SID.

2.8. MPLS Label Downloaded to FIB for Global and Local SIDs

The label corresponding to the global SID "Si" represented by the global index "I" downloaded to FIB is used to match packets whose active segment (and hence topmost label) is "Si". The value of this label is calculated as specified in Section 2.4.

For Local SIDs, the MCC is responsible for downloading the correct label value to FIB. For example, an IGP with SR extensions I-D.ietf-isis-segment-routing-extensions, I-D.ietf-ospf-segment-routing-extensions] allocates and downloads the MPLS label corresponding to an Adj-SID [I-D.ietf-spring-segment-routing].

2.9. Active Segment

When instantiated in the MPLS domain, the active segment on a packet corresponds to the topmost label on the packet that is calculated
according to the procedure specified in Sections 2.10 and 2.11. When arriving at a node, the topmost label corresponding to the active SID matches the MPLS label downloaded to FIB as specified in Section 2.4.

2.10. Forwarding behavior for Global SIDs

This section specifies forwarding behavior, including the calculation of outgoing labels, that corresponds to a global SID when applying PUSH, CONTINUE, and NEXT operations in the MPLS forwarding plane. This document covers the calculation of the outgoing label for the top label only. The case where the outgoing label is not the top label and is part of a stack of labels that instantiates a routing policy or a traffic engineering tunnel is covered in other documents such as [I.D.filsfils-spring-segment-routing-policy].

2.10.1. Forwarding for PUSH and CONTINUE of Global SIDs

Suppose an MCC on a router "R0" determines that PUSH or CONTINUE operation is to be applied to an incoming packet whose active SID is the global SID "Si" represented by the global index "I" and owned by the router Ri before sending the packet towards a neighbor "N" directly connected to "R0" through a physical or virtual interface such as UDP tunnel [RFC7510] or L2TPv3 tunnel [RFC4817].

The method by which the MCC on router "R0" determines that PUSH or CONTINUE operation must be applied using the SID "Si" is beyond the scope of this document. An example of a method to determine the SID "Si" for PUSH operation is the case where IS-IS [I-D.ietf-isis-segment-routing-extensions] receives the prefix-SID "Si" sub-TLV advertised with prefix "P/m" in TLV 135 and the destination address of the incoming IPv4 packet is covered by the prefix "P/m".

For CONTINUE operation, an example of a method to determine the SID "Si" is the case where IS-IS [I-D.ietf-isis-segment-routing-extensions] receives the prefix-SID "Si" sub-TLV advertised with prefix "P" in TLV 135 and the top label of the incoming packet matches the MPLS label in FIB corresponding to the SID "Si" on the router "R0".

The forwarding behavior for PUSH and CONTINUE corresponding to the SID "Si"

- If the neighbor "N" does not support SR or "I" does not satisfy the inequality specified in Section 2.4 for the SRGB of the neighbor "N"
o If it is possible to send the packet towards the neighbor "N" using standard MPLS forwarding behavior as specified in [RFC3031] and [RFC3032], then forward the packet. The method by which a router decides whether it is possible to send the packet to "N" or not is beyond the scope of this document. For example, the router "R0" can use the downstream label determined by another MCC, such as LDP [RFC5036], to send the packet.

o Else if there are other usable next-hops, then use other next-hops to forward the incoming packet. The method by which the router "R0" decides on the possibility of using other next-hops is beyond the scope of this document. For example, the MCC on "R0" may chose to send an IPv4 packet without pushing any label to another next-hop.

o Otherwise drop the packet.

o Else

  o Calculate the outgoing label as specified in Section 2.4 using the SRGB of the neighbor "N"

  o If the operation is PUSH

    . Push the calculated label according the MPLS label pushing rules specified in [RFC3032]

  o Else

    . swap the incoming label with the calculated label according to the label swapping rules in [RFC3032]

  o Send the packet towards the neighbor "N"

2.10.2. Forwarding for NEXT Operation for Global SIDs

As specified in Section 2.7.3 NEXT operation corresponds to popping the top most label. The forwarding behavior is as follows

  o Pop the topmost label

  o Apply the instruction associated with the incoming label that has been popped
The action on the packet after popping the topmost label depends on the instruction associated with the incoming label as well as the contents of the packet right underneath the top label that got popped. Examples of NEXT operation are described in Section 3.

2.11. Forwarding Behavior for Local SIDs

This section specifies the forwarding behavior for local SIDs when SR is instantiated over the MPLS forwarding plane.

2.11.1. Forwarding for PUSH Operation on Local SIDs

Suppose an MCC on a router "R0" determines that PUSH operation is to be applied to an incoming packet using the local SID "Si" before sending the packet towards a neighbor "N" directly connected to R0 through a physical or virtual interface such as UDP tunnel [RFC7510] or L2TPv3 tunnel [RFC4817].

An example of such local SID is an Adj-SID allocated and advertised by IS-IS [I-D.ietf-isis-segment-routing-extensions]. The method by which the MCC on "R0" determines that PUSH operation is to be applied to the incoming packet is beyond the scope of this document. An example of such method is backup path used to protect against a failure using TI-LFA [I.D.bashandy-rtgwg-segment-routing-ti-lfa].

As mentioned in [I-D.ietf-spring-segment-routing], a local SID is specified by an MPLS label. Hence the PUSH operation for a local SID is identical to label push operation [RFC3032] using any MPLS label. The forwarding action after pushing the MPLS label corresponding to the local SID is also determined by the MCC. For example, if the PUSH operation was done to forward a packet over a backup path calculated using TI-LFA, then the forwarding action may be sending the packet to a certain neighbor that will in turn continue to forward the packet along the backup path.

2.11.2. Forwarding for CONTINUE Operation for Local SIDs

A local SID on a router "R0" corresponds to a local label such as an Adj-SID. In such scenario, the outgoing label towards a next-hop "N" is determined by the MCC running on the router "R0" and the forwarding behavior for CONTINUE operation is identical to swap operation [RFC3032] on an MPLS label.

2.11.3. Outgoing label for NEXT Operation for Local SIDs

NEXT operation for Local SIDs is identical to NEXT operation for global SIDs specified in Section 2.10.2.
3. IGP Segments Examples

Consider the network diagram of Figure 1 and the IP address and IGP Segment allocation of Figure 2. Assume that the network is running IS-IS with SR extensions [I-D.ietf-isis-segment-routing-extensions] and all links have the same metric. The following examples can be constructed.

```
        +--------+
       /          \
R0-----R1----R2---------R3-----R8
 \    \       /     \
  +---R4+-     /        \
     +-----R5-----+
```

Figure 1: IGP Segments - Illustration
3.1. Example 1

Suppose R1 wants to send an IPv4 packet P1 to R8. In this case, R1 needs to apply PUSH operation to the IPv4 packet.

Remember that the SID index "8" is a global IGP segment attached to the IP prefix 192.0.2.8/32. Its semantic is global within the IGP domain: any router forwards a packet received with active segment 8 to the next-hop along the ECMP-aware shortest-path to the related prefix.

R2 is the next-hop along the shortest path towards R8. By applying the steps in Section 2.8 the local label downloaded to R1’s FIB corresponding to the global SID index 8 is 1008 because the SRGB of R2 is [1000,5000] as shown in Figure 2.
Because the packet is IPv4, R1 applies the PUSH operation using the label value 1008 as specified in Section 2.10.1. The resulting MPLS header will have the "S" bit [RFC3032] set because it is followed directly by an IPv4 packet.

The packet arrives at router R2. Because the top label 1008 corresponds to the IGP SID "8", which is the prefix-SID attached to the prefix 192.0.2.8/32 owned by the R8, then the instruction associated with the SID is "forward the packet using all ECMP/UCMP interfaces and all ECMP/UCMP next-hop(s) along the shortest path towards R8". Because R2 is not the penultimate hop, R2 applies the CONTINUE operation to the packet and sends it to R3 using one of the two links connected to R3 with top label 1008 as specified in Section 2.10.1.

R3 receives the packet with top label 1008. Because the top label 1008 corresponds to the IGP SID "8", which is the prefix-SID attached to the prefix 192.0.2.8/32 owned by the R8, then the instruction associated with the SID is "send the packet using all ECMP interfaces and all next-hop(s) along the shortest path towards R8". Because R3 is the penultimate hop, R3 applies NEXT operation then sends the packet to R8. The NEXT operation results in popping the outer label and sending the packet as a pure IPv4 packet to R8. The

In conclusion, the path followed by P1 is R1-R2--R3-R8. The ECMP-awareness ensures that the traffic be load-shared between any ECMP path, in this case the two north and south links between R2 and R3.

3.2. Example 2

Suppose the right most router R0 wants to send a packet P2 to R8 over the path <R2, (north link between R2 and R3)>. In that case, the router R0 needs to use the SID list <2, 9001, 8>. Using the calculation techniques specified in Section 2.10 and 2.11 the resulting label stack starting from the topmost label is <1002, 9001, 1008>.

The MPLS label 1002 is the MPLS instantiation of the global IGP segment index 2 attached to the IP prefix 192.0.2.2/32. Its semantic is global within the IGP domain: any router forwards a packet received with active segment 1002 to the next-hop along the shortest-path to the related prefix.

The MPLS label 9001 is a local IGP segment attached by node R2 to its north link to R3. Its semantic is local to node R2: R2 applies NEXT operation, which corresponding to popping the outer label, then
switches a packet received with active segment 9001 towards the north link to R3.

In conclusion, the path followed by P2 is R0-R1-R2-north-link-R3-R8.

3.3. Example 3

R0 may send a packet P3 along the same exact path as P2 using a different segment list <2,9003,8> which corresponds to the label stack <1002, 9003, 1008>.

9003 is a local IGP segment attached by node R2 to both its north and south links to R3. Its semantic is local to node R2: R2 applies NEXT operation, which corresponds to popping the top label, then switches a packet received with active segment 9003 towards either the north or south links to R3 (e.g. per-flow loadbalancing decision).

In conclusion, the path followed by P3 is R0-R1-R2-any-link-R3-R8.

3.4. Example 4

R0 may send a packet P4 to R8 while avoiding the links between R2 and R3 by pushing the SID list <4,8>, which corresponds to the label stack <1004, 1008>.

1004 is a global IGP segment attached to the IP prefix 192.0.2.4/32. Its semantic is global within the IGP domain: any router forwards a packet received with active segment 1004 to the next-hop along the shortest-path to the related prefix.

In conclusion, the path followed by P4 is R0-R1-R2-R4-R3-R8.

3.5. Example 5

R0 may send a packet P5 to R8 while avoiding the links between R2 and R3 and still benefiting from all the remaining shortest paths (via R4 and R5) by pushing the SID list <1009, 8> which corresponds to the label stack <2009, 1008> using the steps specified in Sections 2.10 and 2.11.

1009 is a global anycast-SID [I-D.ietf-spring-segment-routing] attached to the anycast IP prefix 198.51.100.9/32. Its semantic is global within the IGP domain: any router forwards a packet received with top label 2009 (corresponding to the active segment 1009) to the next-hop along the shortest-path to the related prefix.
In conclusion, the path followed by P5 is either R0-R1-R2-R4-R3-R8 or R0-R1-R2-R5-R3-R8.

4. IANA Considerations

This document does not make any request to IANA.

5. Manageability Considerations

This document describes the applicability of Segment Routing over the MPLS data plane. Segment Routing does not introduce any change in the MPLS data plane. Manageability considerations described in [I-D.ietf-spring-segment-routing] applies to the MPLS data plane when used with Segment Routing. SR OAM use cases for the MPLS data plane are defined in [I-D.ietf-spring-oam-usecase]. SR OAM procedures for the MPLS data plane are defined in [RFC8287].

6. Security Considerations

This document does not introduce additional security requirements and mechanisms other than the ones described in [I-D.ietf-spring-segment-routing].

7. Contributors

The following contributors have substantially helped the definition and editing of the content of this document:
8. Acknowledgements

The authors would like to thank Les Ginsberg and Himanshu Shah for their comments on this document.

This document was prepared using 2-Word-v2.0.template.dot.

9. References

9.1. Normative References


9.2. Informative References


Authors' Addresses

Ahmed Bashandy
Individual

Email: abashandy.ietf@gmail.com

Clarence Filsfils (editor)
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Stefano Previdi (editor)
Cisco Systems, Inc.
Italy

Email: stefano@previdi.net

Bruno Decraene
Orange
FR

Email: bruno.decraene@orange.com

Stephane Litkowski
Orange
FR

Email: stephane.litkowski@orange.com

Rob Shakir
Google
US

Email: robjs@google.com
Bidirectional Forwarding Detection (BFD) in Segment Routing Networks
Using MPLS Dataplane
draft-mirsky-spring-bfd-05

Abstract

Segment Routing (SR) architecture leverages the paradigm of source routing. It can be realized in the Multiprotocol Label Switching (MPLS) network without any change to the data plane. A segment is encoded as an MPLS label and an ordered list of segments is encoded as a stack of labels. Bidirectional Forwarding Detection (BFD) is expected to monitor any kind of paths between systems. This document defines how to use Label Switched Path Ping to bootstrap and control path in reverse direction of a BFD session on the Segment Routing static MPLS tunnel and applicability of BFD Demand mode to SR-MPLS domain.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 2, 2018.
Internet-Draft             BFD in SPRING MPLS                 March 2018

1.  Introduction

[RFC5880], [RFC5881], and [RFC5883] established the Bidirectional Forwarding Detection (BFD) protocol for IP networks. [RFC5884] and [RFC7726] set rules of using BFD Asynchronous mode over Multiprotocol Label Switching (MPLS) Label Switched Path (LSP). These latter standards implicitly assume that the egress BFD peer, which is the egress Label Edge Router (LER), will use the shortest path route regardless of the path the ingress LER uses to send BFD control packets towards it.
This document defines use of LSP Ping for Segment Routing networks over MPLS dataplane [RFC8287] to bootstrap and control path of a BFD session from the egress to ingress LER using static MPLS tunnel.

1. Conventions used in this document

1.1. Terminology

BFD: Bidirectional Forwarding Detection
FEC: Forwarding Equivalence Class
MPLS: Multiprotocol Label Switching
SR-MPLS Segment Routing with MPLS data plane
LSP: Label Switching Path
LER: Label Edge Router

1.2. Requirements Language

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

2. Bootstrapping BFD session over Segment Routed tunnel

As demonstrated in [RFC8287] introduction of Segment Routing network domains with an MPLS data plane requires three new sub-TLVs that MAY be used with Target Forwarding Equivalence Class (FEC) TLV. Section 6.1 addresses use of the new sub-TLVs in Target FEC TLV in LSP ping and LSP traceroute. For the case of LSP ping the [RFC8287] states that:

Initiator MUST include FEC(s) corresponding to the destination segment.

Initiator, i.e. ingress LSR, MAY include FECs corresponding to some or all of segments imposed in the label stack by the ingress LSR to communicate the segments traversed.

It has been noted in [RFC5884] that a BFD session monitors for defects particular <MPLS LSP, FEC> tuple. [RFC7726] clarified how to
establish and operate multiple BFD sessions for the same <MPLS LSP, FEC> tuple. Because only ingress edge router is aware of the SR-based explicit route the egress edge router can associate the LSP ping with BFD Discriminator TLV with only one of the FECs it advertised for the particular segment. Thus this document clarifies that:

When LSP Ping is used to bootstrap a BFD session the FEC corresponding to the destination segment to be associated with the BFD session MUST be as the very last sub-TLV in the Target FEC TLV.

Encapsulation of a BFD Control packet in Segment Routing network with MPLS dataplane MUST follow Section 7 [RFC5884] when IP/UDP header used and MUST follow Section 3.4 [RFC6428] without IP/UDP header being used.

3. Use BFD Reverse Path TLV over Segment Routed MPLS Tunnel

For BFD over MPLS LSP case, per [RFC5884], egress LER MAY send BFD control packet to the ingress LER either over IP network or an MPLS LSP. Similarly, for the case of BFD over p2p segment tunnel with MPLS data plane, the ingress LER MAY route BFD control packet over IP network, as described in [RFC5883], or transmit over a segment tunnel, as described in Section 7 [RFC5884]. In some cases there may be a need to direct egress BFD peer to use specific path for the reverse direction of the BFD session by using the BFD Reverse Path TLV and following all procedures as defined in [I-D.ietf-mpls-bfd-directed].

4. Use Non-FEC Path TLV

For the case of MPLS dataplane, Segment Routing Architecture [I-D.ietf-spring-segment-routing] explains that "a segment is encoded as an MPLS label. An ordered list of segments is encoded as a stack of labels." YANG Data Model for MPLS Static LSPs [I-D.ietf-mpls-static-yang] models outgoing MPLS labels to be imposed as leaf-list [RFC6020], i.e., as array of rt-types:mpls-label [RFC8294].

This document defines new optional Non-FEC Path TLV. The format of the Non-FEC Path TLV is presented in Figure 1.
Non-FEC Path TLV Type is 2 octets in length and has a value of TBD1 (to be assigned by IANA as requested in Section 7.1).

Length field is 2 octets long and defines the length in octets of the Non-FEC Path field.

Non-FEC Path field contains a sub-TLV. Any Non-FEC Path sub-TLV (defined in this document or to be defined in the future) for Non-FEC Path TLV type MAY be used in this field. None or one sub-TLV MAY be included in the Non-FEC Path TLV. If no sub-TLV has been found in the Non-FEC Path TLV, the egress BFD peer MUST revert to using the reverse path selected based on its local policy. If there are more than one sub-TLV, then the Return Code in echo reply MUST be set to value TBD3 "Too Many TLVs Detected" (to be assigned by IANA as requested in Table 4).

Non-FEC Path TLV MAY be used to specify the reverse path of the BFD session identified in the BFD Discriminator TLV. If the Non-FEC Path TLV is present in the echo request message the BFD Discriminator TLV MUST be present as well. If the BFD Discriminator TLV is absent when the Non-FEC Path TLV is included, then it MUST be treated as malformed Echo Request, as described in [RFC8029].

This document defines Static Routing MPLS Tunnel sub-TLV that MAY be used with the Non-FEC Path TLV. The format of the sub-TLV is presented in Figure 2.
The Segment Routing MPLS Tunnel sub-TLV Type is two octets in length, and has a value of TBD2 (to be assigned by IANA as requested in Section 7.1).

The egress LSR MUST use the Value field as label stack for BFD control packets for the BFD session identified by the source IP address of the MPLS LSP Ping packet and the value in the BFD Discriminator TLV. Label Entries MUST be in network order.

5. BFD Reverse Path TLV over Segment Routed MPLS Tunnel with Dynamic Control Plane

When Segment Routed domain with MPLS data plane uses distributed tunnel computation BFD Reverse Path TLV MAY use Target FEC sub-TLVs defined in [RFC8287].

6. Applicability of BFD Demand Mode in SR-MPLS Domain

[I-D.mirsky-bfd-mpls-demand] defines how Demand mode of BFD, specified in sections 6.6 and 6.18.4 of [RFC5880], can be used to monitor uni-directional MPLS LSP. Similar procedures can be following in SR-MPLS to monitor uni-directional SR tunnels:

- ingress SR node bootstraps BFD session over SR-MPLS in Async BFD mode;

- once BFD session is Up, the ingress node switches the egress BFD node into the Demand mode by setting D field in BFD Control packet it transmits;
if the egress BFD node detects the failure of the BFD session, it sends its BFD control packet to the ingress over the IP network with Poll sequence;

- if the ingress node receives BFD control packet from remote node in Demand mode with Poll sequence and Diag field indicating the failure, the ingress transmits BFD control packet with Final over IP and switches the BFD over SR-MPLS back into Async mode, sending BFD Control packets one per second.

7. IANA Considerations

7.1. Non-FEC Path TLV

IANA is requested to assign new TLV type from the Standards Action range of the registry "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" as defined in the Table 1.

<table>
<thead>
<tr>
<th>Value</th>
<th>TLV Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD1</td>
<td>Non-FEC Path TLV</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 1: New Non-FEC Path TLV

IANA is requested to create new Non-FEC Path sub-TLV registry for the Non-FEC Path TLV as described in Table 2.

<table>
<thead>
<tr>
<th>Range</th>
<th>Registration Procedures</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-16383</td>
<td>Standards Action</td>
<td>This range is for mandatory TLVs or for optional TLVs that require an error message if not recognized. Experimental RFC needed</td>
</tr>
<tr>
<td>16384-31743</td>
<td>Specification Required</td>
<td></td>
</tr>
<tr>
<td>32768-49161</td>
<td>Standards Action</td>
<td>This range is for optional TLVs that can be silently dropped if not recognized. Experimental RFC needed</td>
</tr>
<tr>
<td>49162-64511</td>
<td>Specification Required</td>
<td></td>
</tr>
<tr>
<td>64512-65535</td>
<td>Private Use</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Non-FEC Path sub-TLV registry
IANA is requested to allocate following values from the Non-FEC Path sub-TLV registry as defined in Table 3.

```
+-------+-------------------------------------+---------------+
| Value | Description                         | Reference     |
|-------+-------------------------------------+---------------+
| 0     | Reserved                            | This document |
| TBD2  | Segment Routing MPLS Tunnel sub-TLV | This document |
| 65535 | Reserved                            | This document |
```

Table 3: New Segment Routing Tunnel sub-TLV

7.2. Return Code

IANA is requested to create Non-FEC Path sub-TLV subregistry for the new Non-FEC Path TLV. assign a new Return Code value from the "Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters" registry, "Return Codes" sub-registry, as follows using a Standards Action value.

```
+--------+-------------------------+---------------+
| Value  | Description             | Reference     |
|--------+-------------------------+---------------+
| X TBD3 | Too Many TLVs Detected. | This document |
```

Table 4: New Return Code

8. Security Considerations

Security considerations discussed in [RFC5880], [RFC5884], [RFC7726], and [RFC8029] apply to this document.

9. Acknowledgements

TBD

10. References

10.1. Normative References

[MRK.ITF-MPLS-BFD-DIRECTED]
[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.mirsky-bfd-mpls-demand]
Mirsky, G., "BFD in Demand Mode over Point-to-Point MPLS
LSP", draft-mirsky-bfd-mpls-demand-02 (work in progress),
October 2017.

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

(BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,

(BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881,
DOI 10.17487/RFC5881, June 2010,

(BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883,

[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,

"Proactive Connectivity Verification, Continuity Check,
and Remote Defect Indication for the MPLS Transport
Profile", RFC 6428, DOI 10.17487/RFC6428, November 2011,

Aldrin, "Clarifying Procedures for Establishing BFD
Sessions for MPLS Label Switched Paths (LSPs)", RFC 7726,
DOI 10.17487/RFC7726, January 2016,
10.2. Informative References

[I-D.ietf-mpls-static-yang]


Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Jeff Tantsura
Nuage Networks

Email: jefftant.ietf@gmail.com
Abstract

This document describes a YANG data model for Segment Routing IPv6 (SRv6) base. The model serves as a base framework for configuring and managing an SRv6 subsystem and expected to be augmented by other SRv6 technology models accordingly. Additionally, this document also specifies the model for the SRv6 Static application.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."
This Internet-Draft will expire on September 6, 2018.

Copyright Notice

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

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

Table of Contents

1.  Introduction . . . . . . . . . . . . . . . . . . . . . . . .   3
2.  Specification of Requirements . . . . . . . . . . . . . . . .   3
3.  YANG Model . . . . . . . . . . . . . . . . . . . . . . . . . .   4
   3.1.  Overview . . . . . . . . . . . . . . . . . . . . . . . . .   4
   3.2.  SRv6 Types . . . . . . . . . . . . . . . . . . . . . . . .   4
   3.3.  SRv6 Base . . . . . . . . . . . . . . . . . . . . . . . . .   5
      3.3.1.  Configuration . . . . . . . . . . . . . . . . . . . .   5
      3.3.2.  State . . . . . . . . . . . . . . . . . . . . . . . . .   6
      3.3.3.  Notification . . . . . . . . . . . . . . . . . . . .   8
   3.4.  SRv6 Static . . . . . . . . . . . . . . . . . . . . . . . .   9
      3.4.1.  Configuration . . . . . . . . . . . . . . . . . . . .   9
      3.4.2.  State . . . . . . . . . . . . . . . . . . . . . . . . .  13
      3.4.3.  Notification . . . . . . . . . . . . . . . . . . . .   13
4.  Work In Progress . . . . . . . . . . . . . . . . . . . . . . .  14
5.  Pending Items . . . . . . . . . . . . . . . . . . . . . . . . .  14
6.  YANG Specification . . . . . . . . . . . . . . . . . . . . . .  14
   6.1.  SRv6 Types . . . . . . . . . . . . . . . . . . . . . . . .  15
   6.2.  SRv6 Base . . . . . . . . . . . . . . . . . . . . . . . . .  27
   6.3.  SRv6 Static . . . . . . . . . . . . . . . . . . . . . . . .  41
7.  Security Considerations . . . . . . . . . . . . . . . . . . .  58
8.  IANA Considerations . . . . . . . . . . . . . . . . . . . . .  58
9.  Acknowledgments . . . . . . . . . . . . . . . . . . . . . . .  58
10. References . . . . . . . . . . . . . . . . . . . . . . . . . .  58
    10.1.  Normative References . . . . . . . . . . . . . . . . . .  58
    10.2.  Informative References . . . . . . . . . . . . . . . . .  59
Authors’ Addresses . . . . . . . . . . . . . . . . . . . . . . . .  61
1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] is one of the network management protocols that defines mechanisms to manage network devices. YANG [RFC6020] is a modular language that represents data structures in an XML tree format, and is used as a data modeling language for the NETCONF.

Segment Routing (SR), as defined in [I-D.ietf-spring-segment-routing], leverages the source routing paradigm where a node steers a packet through an ordered list of instructions, called segments. SR, thus, allows enforcing a flow through any topological path and/or service chain while maintaining per-flow state only at the ingress nodes to the SR domain. When applied to ipv6 data-plane (i.e. SRv6), SR requires a type of routing header (SRH) in an IPv6 packet that is used to encode an ordered list of IPv6 addresses (SIDs). The active segment is indicated by the Destination Address of the packet, and the next segment is indicated by a pointer in the SRH [I-D.ietf-6man-segment-routing-header]. The various functions and behaviors corresponding to network programming using SRv6 are specified in [I-D.filsfils-spring-srv6-network-programming].

This document introduces a YANG data model for base SRv6 that would serve as a base framework for configuring and managing an SRv6 subsystem. It is expected that other SRv6 technology models (e.g. ISIS, OSPFv3, BGP, EVPN, Service Chaining) will augment this model accordingly. Furthermore, to illustrate basic behaviors as captured in [I-D.filsfils-spring-srv6-network-programming], this document also specifies a YANG model for the SRv6-Static application.

The model currently defines the following constructs that are used for managing SRv6:

- Configuration
- Operational State
- Notifications

2. Specification of Requirements

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. YANG Model

3.1. Overview

This document defines following new YANG modules:

- ietf-srv6-types: defines common and basic types related to SRv6
- ietf-srv6-base: specifies management model for SRv6 base constructs (locator, SIDs, etc.)
- ietf-srv6-static: specifies management model for SRv6-static application

The modeling in this document complies with the Network Management Datastore Architecture (NMDA) [I-D.ietf-netmod-revised-datastores]. The operational state data is combined with the associated configuration data in the same hierarchy [I-D.ietf-netmod-rfc6087bis]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all "config true" (rw) and "config false" (ro) nodes defined in the schema.

3.2. SRv6 Types

SRv6 common types and definitions are defined in the new module "ietf-srv6-types". The main types defined in this module include:

- srv6-sid: SRv6 SID
- srv6-func-opcode: Typedef for FUNC opcode in an SRv6 SID
- srv6-func-opcode-reserved: Typedef for "reserved" FUNC opcode
- srv6-func-opcode-unreserved: Typedef for "unreserved" (allocatable) FUNC opcode
- srv6-func-opcode-reserved-type: Enum (list) of "reserved" FUNC opcode
- srv6-endpoint-type: SRv6 Endpoint behaviors [I-D.filsfils-spring-srv6-network-programming] identity type
- srv6-transit-type: SRv6 Transit behavior types [I-D.filsfils-spring-srv6-network-programming] enumeration
- srv6-security-rule-type: SRv6 Security rule type [I-D.filsfils-spring-srv6-network-programming] enumeration
3.3. SRv6 Base

The base SRv6 model is captured by ietf-srv6-base module. This module augments "/rt:routing" and specifies the configuration, operational state, and notification events required to manage base SRv6.

The associated YANG specification for this module is captured in Section 6.2.

3.3.1. Configuration

The module defines some fundamental items required to configure an SRv6 network:

- SRv6 Enablement: Enable Segment-Routing SRv6 feature
- Encapsulation Parameters: Provide encapsulation related parameters, such as source-address and ip-ttl-propagation, to be used when performing T.Encap*
- Locator(s) Specification: SRv6 locator is a fundamental construct for an SRv6 network. This is the construct from which SID (function opcodes) are allocated that on the local box, and advertised to and used by remote nodes for reachability. A locator is identified by a name and has associated prefix. It is possible to have more than one locator per node. In case of more than one locator, there is one and only one locator designated as the default locator.

Following is a simplified graphical representation of the data model for SRv6 base configuration:  

- srv6-counter-type: SRv6 Counter type
  [I-D.filsfils-spring-srv6-network-programming] enumeration

The associated YANG specification for this module is captured in Section 6.1.
module: ietf-srv6-base
    augment /rt:routing:
        +--rw srv6
        |  +--rw enable?              boolean
        |  +--rw encapsulation
        |     |  +--rw source-address?       inet:ipv6-address
        |     |  +--rw ip-ttl-propagation?   boolean
        |  +--rw locators
        |     +--rw locator* [name]
    |          |  +--rw name                      string
    |          |  +--rw enable?                   boolean
    |          |     +--rw is-default                boolean
    |          |     +--rw prefix
    |          |          +--rw address    inet:ipv6-address
    |          |          +--rw length     srv6-types:srv6-locator-len

Figure 1: SRv6 Base - Config Tree

3.3.2. State

As per NMDA model, the state related to configuration items specified in above section Section 3.3.1 can be retrieved from the same tree. This section defines other operational state items related to SRv6 base.

The operational state corresponding to the SRv6 base includes:

- **node capabilities**: provides information on the node (hardware) capabilities and support regarding various SRv6 aspects and features including end behaviors, transit behaviors, security rules, counter/stats support, and other SRv6 parameters that need to be signaled in an SRv6 network by the protocols.

- **locator**: provides information related to a locator. The information includes locator operational state, and state of address conflict with any ipv6 address configured on local interfaces etc.

- **local-sid**: provides information related to local-SIDs allocated and/or installed on the node. This includes two types of information:
  1. aggregate across all local-SIDs such as aggregate counters
  2. per local-SID information such as allocation type (dynamic or explicit), SID owner protocol(s)/client(s), forwarding [paths] information, and stats/counters.
Following is a simplified graphical representation of the data model for the SRv6 operational state:

```yang
module: ietf-srv6-base
augment /rt:routing:
  +--rw srv6
    +--rw locators
      +--rw locator* [name]
        +--rw name                      string
        +--ro operational-status?       srv6-types:srv6-status-type
        +--ro is-in-address-conflict?   boolean
    +--ro node-capabilities
      +--ro end-behavior* [type]
        |  +--ro type         identityref
        |  +--ro supported    boolean
      +--ro transit-behavior* [type]
        |  +--ro type         srv6-types:srv6-transit-type
        |  +--ro supported    boolean
      +--ro signaled-parameters
        |  +--ro max-sl?            uint8
        |  +--ro max-end-pop-srh?   uint8
        |  +--ro max-t_insert?      uint8
        |  +--ro max-t_encap?       uint8
        |  +--ro max-end_d?         uint8
      +--ro security-rule* [type]
        |  +--ro type         srv6-types:srv6-security-rule-type
        |  +--ro supported    boolean
      +--ro counters* [type]
        |  +--ro type         srv6-types:srv6-counter-type
        |  +--ro supported    boolean
    +--ro local-sids
      +--ro counters
        |  +--ro cnt3
        |      +--ro in-pkts?     yang:counter64
        |      +--ro in-octets?  yang:counter64
      +--ro local-sid* [sid]
        |  +--ro sid                  srv6-types:srv6-sid
        |  +--ro is-reserved?         boolean
        |  +--ro end-behavior-type?   identityref
        |  +--ro alloc-type?          srv6-types:sid-alloc-type
        |  +--ro owner* [type instance]
        |      +--ro type         identityref
        |      +--ro instance       string
        |      +--ro is-winner?      boolean
        |  +--ro forwarding
        |      +--ro is-installed?   boolean
```

This model defines a list of notifications to inform an operator of important events detected during the SRv6 operation. These events include events related to:

- locator operational state changes
- local-SID collision event

Following is a simplified graphical representation of the data model for SRv6 notifications:
module: ietf-srv6-base

notifications:

   +---n srv6-locator-status-event
       |   +--ro operational-status?   srv6-types:srv6-status-type

   +---n srv6-sid-collision-event
       +--ro sid?         srv6-types:srv6-sid
       |   +--ro existing
       |       +--ro end-behavior-type?   identityref
       |   +--ro requested
       |       +--ro end-behavior-type?   identityref

Figure 3: SRv6 Base - Notification Tree

3.4. SRv6 Static

SRv6-Static application allows a user to specify SRv6 local SIDs and program them in the forwarding plane. The SRv6-Static model is captured in the ietf-srv6-static module.

The associated YANG specification for this module is captured in Section 6.3.

3.4.1. Configuration

The SRv6-Static configuration augments the SRv6-base locator tree "/rt:routing/srv6:srv6/locators/srv6:locator"

Following are salient features of SRv6-Static config model:

- Allows static (explicit) configuration for local-SIDs under a given locator
- Given that entry is scoped under a locator, the key for each entry is function "opcode"
- A user must also specify end-behavior type (End* function) associated with the entry
- A user must also specify behavior-specific data with each entry. For example, for any end behavior requiring a table lookup, a lookup-table need be provided. Similarly, for any end behavior with forwarding next-hops need to specify next-hop information. The example of former include End, End.I, End.DT4, End.DT6, and
Each local-SID entry has zero or more forwarding paths specified.

A forwarding path has next-hop type that depends on the end behavior, and could be either ipv6, or ipv4, or mpls, or l2 type. For example, End.X, End.DX4, End.DX6, End.B6, End.BM, and End.DX2 will have ipv6, ipv4, ipv6, ipv6, mpls, and l2 next-hop types respectively.

For each forwarding next-hop type, the appropriate path attributes are to be specified as well. For L2 type, the only other information required is the L2 interface name. Whereas for L3 (ipv6, ipv4, mpls) types, the information includes L3 interface name, next-hop IP address, weight, and protection information.

Depending on the end behavior type, a forwarding path may have either MPLS or SRv6 encapsulation -- i.e., Stack of out-labels or Stack of SRv6 out-SIDs. The example of former is End.BM and example of later include the rest (End.X, End.DX4/DX6, End.B6 etc.).

Following is a simplified graphical representation of the data model for SRv6 Static configuration:

```yaml
module: ietf-srv6-static
augment /rt:routing/srv6:srv6/locators/srv6:locator:
  +--rw static
    +--rw local-sids
      +--rw sid* [opcode]
        +--rw opcode        srv6-types:srv6-func-opcode-unreserved
        +--rw end-behavior-type identityref
        +--rw end
        +--rw end_psp
        +--rw end_usp
        +--rw end_psp_usp
        +--rw end-t
          | +--rw lookup-table-ipv6    srv6-types:table-id
          +--rw end_t_psp
          | +--rw lookup-table-ipv6    srv6-types:table-id
          +--rw end_t_usp
          | +--rw lookup-table-ipv6    srv6-types:table-id
          +--rw end_t_psp_usp
          | +--rw lookup-table-ipv6    srv6-types:table-id
          +--rw end-x
            | +--rw protected?  boolean
```

End.DT46, whereas example of later include End.X, End.DX4, End.DX6, End.B6, End.BM etc.
++-rw paths
  ++-rw path* [path-index]
    ++-rw path-index           uint8
    ++-rw interface?           if:interface-ref
    ++-rw next-hop?            inet:ipv6-address
    ++-rw weight?              uint32
    ++-rw role?                enumeration
    ++-rw backup-path-index?   uint8
    ++-rw encap
      ++-rw out-sid* [sid]
        ++-rw sid    srv6-types:srv6-sid
++-rw end-x_psp
  ++-rw protected?   boolean
++-rw paths
  ++-rw path* [path-index]
    ++-rw path-index           uint8
    ++-rw interface?           if:interface-ref
    ++-rw next-hop?            inet:ipv6-address
    ++-rw weight?              uint32
    ++-rw role?                enumeration
    ++-rw backup-path-index?   uint8
    ++-rw encap
      ++-rw out-sid* [sid]
        ++-rw sid    srv6-types:srv6-sid
++-rw end-x_usp
  ++-rw protected?   boolean
++-rw paths
  ++-rw path* [path-index]
    ++-rw path-index           uint8
    ++-rw interface?           if:interface-ref
    ++-rw next-hop?            inet:ipv6-address
    ++-rw weight?              uint32
    ++-rw role?                enumeration
    ++-rw backup-path-index?   uint8
    ++-rw encap
      ++-rw out-sid* [sid]
        ++-rw sid    srv6-types:srv6-sid
++-rw end-x_psp_usp
  ++-rw protected?   boolean
++-rw paths
  ++-rw path* [path-index]
    ++-rw path-index           uint8
    ++-rw interface?           if:interface-ref
    ++-rw next-hop?            inet:ipv6-address
    ++-rw weight?              uint32
    ++-rw role?                enumeration
    ++-rw backup-path-index?   uint8
    ++-rw encap

Internet-Draft  YANG Data Model for SRv6  March 2018

---rw out-sid* [sid]
  +++-rw sid  srv6-types:srv6-sid

---rw end-b6
  +++-rw policy-name  string
  +++-rw paths
    +++-rw path* [path-index]
      +++-rw path-index  uint8
      +++-rw interface?  if:interface-ref
      +++-rw next-hop?   inet:ipv6-address
      +++-rw weight?     uint32
      +++-rw role?       enumeration
      +++-rw backup-path-index?  uint8
    +++-rw encap
      +++-rw out-sid* [sid]
        +++-rw sid  srv6-types:srv6-sid

++-rw end-b6-encaps
  +++-rw policy-name  string
  +++-rw source-address  inet:ipv6-address
  +++-rw paths
    +++-rw path* [path-index]
      +++-rw path-index  uint8
      +++-rw interface?  if:interface-ref
      +++-rw next-hop?   inet:ipv6-address
      +++-rw weight?     uint32
      +++-rw role?       enumeration
      +++-rw backup-path-index?  uint8
    +++-rw encap
      +++-rw out-sid* [sid]
        +++-rw sid  srv6-types:srv6-sid

++-rw end-bm
  +++-rw policy-name  string
  +++-rw paths
    +++-rw path* [path-index]
      +++-rw path-index  uint8
      +++-rw interface?  if:interface-ref
      +++-rw next-hop?   inet:ip-address
      +++-rw weight?     uint32
      +++-rw role?       enumeration
      +++-rw backup-path-index?  uint8
    +++-rw encap
      +++-rw out-label* [label]
        +++-rw label  rt-types:mpls-label

++-rw end-dx6
  +++-rw paths
    +++-rw path* [path-index]
      +++-rw path-index  uint8
      +++-rw interface?  if:interface-ref
      +++-rw next-hop?   inet:ipv6-address
Internet-Draft          YANG Data Model for SRv6              March 2018

---rw weight?          uint32
---rw role?            enumeration
---rw backup-path-index?  uint8
---rw encap
   ---rw out-sid* [sid]
      ---rw sid       srv6-types:srv6-sid
---rw end-dx4
   ---rw paths
      ---rw path* [path-index]
         ---rw path-index  uint8
         ---rw interface?  if:interface-ref
         ---rw next-hop?   inet:ipv4-address
         ---rw weight?     uint32
         ---rw role?       enumeration
         ---rw backup-path-index?  uint8
         ---rw encap
            ---rw out-sid* [sid]
               ---rw sid       srv6-types:srv6-sid
---rw end-dt6
   ---rw lookup-table-ipv6  srv6-types:table-id
---rw end-dt4
   ---rw lookup-table-ipv4  srv6-types:table-id
---rw end-dt46
   ---rw lookup-table-ipv4  srv6-types:table-id
   ---rw lookup-table-ipv6  srv6-types:table-id
---rw end-dx2
   ---rw paths
      ---rw interface  if:interface-ref
---rw end-otp

Figure 4: SRv6 Static - Config Tree

3.4.2. State

As per NMDA model, the state related to configuration items specified in above section Section 3.4.1 can be retrieved from the same tree. The state regarding the local-SIDs created by SRv6-static model can be obtained using the state model of SRv6-base. Hence, there is no additional state identified at this time for SRv6-static.

3.4.3. Notification

None.

4. Work In Progress

The YANG modeling for the corresponding SRv6 protocols and technologies is in progress as follows:

- SRv6 ISIS: Transport SIDs, TILFA, and Microloop Avoidance [I-D.bashandy-isis-srv6-extensions]
- SRv6 TE: SR Policy [I-D.filips-spring-segment-routing-policy]
- SRv6 BGP: L3VPN and Global Table [I-D.dawra-idr-srv6-vpn]
- SRv6 EVPN [I-D.dawra-idr-srv6-vpn]
- SRv6 Service Chaining [I-D.xuclad-spring-sr-service-chaining]
- SRv6 Mobile User-Plane [I-D.ietf-dmm-srv6-mobile-uplane]

The corresponding models will be specified later either in this or in a separate document.

5. Pending Items

Following are the items that will be closed in next revisions:

- Align SRv6 base with SR (MPLS) model [I-D.ietf-spring-sr-yang].
- Extend local-SID collision event/notification in SRv6-base model.
- Add RPC support in the SRv6-base model.
- Add EVPN End functions in the SRv6-Static model.
- Add Service Chaining End functions in the SRv6-Static model.
- Add ARGS support in the SRv6-Static model.
- QoS support

6. YANG Specification

Following are actual YANG definition for SRv6 modules defined earlier in the document.
6.1. SRv6 Types

<CODE BEGINS> file "ietf-srv6-types@2018-03-01.yang" -->

module ietf-srv6-types {
  namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-types";
  prefix srv6-types;

  import ietf-inet-types {
    prefix inet;
  }

  organization "IETF SPRING Working Group";
  contact "WG Web: <http://tools.ietf.org/wg/spring/>
          WG List: <mailto:spring@ietf.org>
          Editor: Kamran Raza
                  <mailto:skraza@cisco.com>
          Editor: Jaganbabu Rajamanickam
                  <mailto:jrajaman@cisco.com>
          Editor: Xufeng Liu
                  <mailto:Xufeng_Liu@jabil.com>
          Editor: Zhibo Hu
                  <mailto:huzhibo@huawei.com>
          Editor: Iftekhar Hussain
                  <mailto:IHussain@infinera.com>
          Editor: Himanshu Shah
                  <mailto:hshah@ciena.com>
          Editor: Daniel Voyer
                  <mailto:daniel.voyer@bell.ca>
          Editor: Hani Elmalky
                  <mailto:hani.elmalky@ericsson.com>
          Editor: Satoru Matsushima
                  <mailto:satoru.matsushima@gmail.com>
          Editor: Katsuhiro Horiba

This YANG module defines the essential types for the management of Segment-Routing with IPv6 dataplane (SRv6).

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).

reference "RFC XXXX";

revision 2018-03-01 {
  description
    "Updated to align with SRv6 network programming draft rev 04";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
}

revision 2017-11-12 {
  description
    "Initial revision";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
}

identity srv6-endpoint-type {
  description
    "Base identity from which specific SRv6 Endpoint types are derived.";
}

/* Endpoints defined under draft-filsfils-spring-srv6-network-programming */

identity End {
  base srv6-endpoint-type;
  description
    "End function (variant: no PSP, no USP).";

identity End_PSP {
  base srv6-endpoint-type;
  description
    "End function (variant: PSP only).";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End_USP {
  base srv6-endpoint-type;
  description
    "End function (variant: USP only).";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End_PSP_USP {
  base srv6-endpoint-type;
  description
    "End function (variant: PSP and USP).";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.X {
  base srv6-endpoint-type;
  description
    "Endpoint with cross-connect to an array
    of layer-3 adjacencies (variant: no PSP, no USP).";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.X_PSP {
  base srv6-endpoint-type;
  description
    "Endpoint with cross-connect to an array
    of layer-3 adjacencies (variant: PSP only).";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.X_USP {
  base srv6-endpoint-type;
description
"Endpoint with cross-connect to an array
of layer-3 adjacencies (variant: USP only).";
reference
"draft-filsfils-spring-srv6-network-programming-04";
}

identity End.X_PSP_USP {
  base srv6-endpoint-type;
  description
  "Endpoint with cross-connect to an array
  of layer-3 adjacencies (variant: PSP and USP).";
  reference
  "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.T {
  base srv6-endpoint-type;
  description
  "Endpoint with specific IPv6 table lookup
  (variant: no PSP, no USP).";
  reference
  "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.T_PSP {
  base srv6-endpoint-type;
  description
  "Endpoint with specific IPv6 table lookup
  (variant: PSP only).";
  reference
  "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.T_USP {
  base srv6-endpoint-type;
  description
  "Endpoint with specific IPv6 table lookup
  (variant: USP only).";
  reference
  "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.T_PSP_USP {
  base srv6-endpoint-type;
  description
  "Endpoint with specific IPv6 table lookup
  (variant: PSP and USP).";
}
reference
"draft-filsfils-spring-srv6-network-programming-04";
}

identity End.B6 {
  base srv6-endpoint-type;
  description
    "Endpoint bound to an SRv6 Policy";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.B6.Encaps {
  base srv6-endpoint-type;
  description
    "This is a variation of the End.B6 behavior where the SRv6 Policy also includes an IPv6 Source Address A.";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.BM {
  base srv6-endpoint-type;
  description
    "Endpoint bound to an SR-MPLS Policy";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.DX6 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and cross-connect to an array of IPv6 adjacencies";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.DX4 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and cross-connect to an array of IPv4 adjacencies";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}
identity End.DT6 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific IPv6 table lookup";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.DT4 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific IPv4 table lookup";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.DT46 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific IP (IPv4 or IPv6) table lookup";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.DX2 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and Layer-2 cross-connect to an L2 interface";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.DX2V {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific VLAN L2 table lookup";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.DT2U {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific IPv4 table lookup";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}
"Endpoint with decapsulation and specific unicast MAC L2 table lookup";
reference
"draft-filsfils-spring-srv6-network-programming-04";
}

identity End.DT2M {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific L2 table flooding";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.OTP {
  base srv6-endpoint-type;
  description
    "Endpoint for OAM operation of timestamp and punt";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

identity End.S {
  base srv6-endpoint-type;
  description
    "Endpoint in search of a target in table TE";
  reference
    "draft-filsfils-spring-srv6-network-programming-04";
}

/* Endpoints defined under draft-xuclad-spring-sr-service-chaining */

identity End.AS {
  base srv6-endpoint-type;
  description
    "Service-Chaining Static proxy for inner type (Ethernet, IPv4 or IPv6)";
  reference
    "draft-xuclad-spring-sr-service-chaining-01";
}

identity End.AD {
  base srv6-endpoint-type;
  description
    "Service-Chaining Dynamic proxy for inner type (Ethernet, IPv4 or IPv6)";
  reference

"draft-xuclad-spring-sr-service-chaining-01";
}

identity End.ASM {
  base srv6-endpoint-type;
  description
    "Service-Chaining Shared memory SR proxy for inner type
     (Ethernet, IPv4 or IPv6)";
  reference
    "draft-xuclad-spring-sr-service-chaining-01";
}

identity End.AM {
  base srv6-endpoint-type;
  description
    "Service-Chaining Masquerading SR proxy";
  reference
    "draft-xuclad-spring-sr-service-chaining-01";
}

/* Endpoints defined under draft-ietf-dmm-srv6-mobile-uplane */

identity End.MAP {
  base srv6-endpoint-type;
  description
    "DMM End.MAP";
  reference
    "draft-ietf-dmm-srv6-mobile-uplane-01";
}

identity End.M.GTP6.UP {
  base srv6-endpoint-type;
  description
    "DMM End.M.GTP6.UP";
  reference
    "draft-ietf-dmm-srv6-mobile-uplane-01";
}

identity End.M.GTP6.DN {
  base srv6-endpoint-type;
typedef srv6-transit-type {
    type enumeration {
        /* draft-filsfils-spring-srv6-network-programming-04 */
        enum T { value 1; description "Transit behavior"; }
        enum T.Insert {
            description "Transit behavior with insertion of an SRv6 policy";
        }
        enum T.Insert.Red {
            description "Transit behavior with reduced insertion of an SRv6 policy";
        }
        enum T.Encaps {
            description "Transit behavior with encap of an SRv6 policy";
        }
        enum T.Encaps.Red {
            description "Transit behavior with reduced encap of an SRv6 policy";
        }
        enum T.Encaps.L2 {
            description "T.Encaps behavior on the received L2 frame";
        }
        enum T.Encaps.L2.Red {
            description "T.Encaps.Red behavior on the received L2 frame";
        }
    }
    description "SRv6 Transit behavior types";
}
typedef srv6-security-rule-type {
    type enumeration {
        /* draft-filsfils-spring-srv6-network-programming-04 */
        enum SEC1 { value 1; description "Security rule SEC1"; }
        enum SEC2 { description "Security rule SEC2"; }
        enum SEC3 { description "Security rule SEC3"; }
        enum SEC4 { description "Security rule SEC4"; }
    }
    description "SRv6 Security rule types";
}

typedef srv6-counter-type {
    type enumeration {
        /* draft-filsfils-spring-srv6-network-programming-04 */
        enum CNT1 { value 1; description "CNT1"; }
        enum CNT2 { description "CNT2"; }
        enum CNT3 { description "CNT3"; }
    }
    description "SRv6 counter types";
}

typedef srv6-sid {
    type inet:ipv6-prefix;
    description "This type defines a SID value in SRv6";
}

typedef srv6-func-opcode {
    type uint32;
    description "This is a typedef for SID FUNC’s opcode type";
}

typedef srv6-func-opcode-reserved {
    type uint32 {
        range "1 .. 63";
    }
    description "This is a typedef for SID FUNC’s reserved opcode type";
}

typedef srv6-func-opcode-unreserved {
    type uint32 {

typedef srv6-func-opcode-reserved-type {
    type enumeration {
        enum invalid {  value 0; description "Invalid opcode"; }
    }
    description "SRv6 SID FUNC Reserved Opcodes";
}

typedef srv6-locator-len {
    type uint8 {
        range "32 .. 96";
    }
    description "This type defines an SRv6 locator len with range constraints";
}

typedef srv6-sid-pfxlen {
    type uint8 {
        range "33 .. 128";
        default 128;
    }
    description "This type defines a SID prefixlen with range constraints";
}

typedef sid-alloc-type {
    type enumeration {
        enum Dynamic {
            description "SID allocated dynamically.";
        }
        enum Explicit {
            description "SID allocated with explicit (static) value";
        }
    }
    description "Types of sid allocation used.";
}

identity srv6-sid-owner-type {

description "Base identity from which SID owner types are derived."
}

identity isis {
    base srv6-sid-owner-type;
    description "ISIS";
}

identity ospfv3 {
    base srv6-sid-owner-type;
    description "OSPFv3";
}

identity bgp {
    base srv6-sid-owner-type;
    description "BGP";
}

identity evpn {
    base srv6-sid-owner-type;
    description "EVPN";
}

identity sr-policy {
    base srv6-sid-owner-type;
    description "SR Policy";
}

identity service-function {
    base srv6-sid-owner-type;
    description "SF";
}

// TODO: Rtg module
typedef table-id {
    type uint32;
    description "Routing Table Id";
}

typedef srv6-status-type {
    type enumeration {
        enum up { value 1; description "State is Up"; }
        enum down { description "State is Down"; }
    }
    description "Status type";
typedef srv6-nexthop-type {
    type enumeration {
        enum ipv4 { value 1; description "IPv4 next-hop"; }
        enum ipv6 { description "IPv6 next-hop"; }
        enum mpls { description "MPLS next-hop"; }
        enum l2 { description "L2 next-hop"; }
    }
    description "Forwarding Next-hop type";
}

} // module

<CODE ENDS>

Figure 5: ietf-srv6-types.yang

6.2. SRv6 Base

<CODE BEGINS> file "ietf-srv6-base@2018-03-01.yang" -->

module ietf-srv6-base {
    prefix srv6;

    import ietf-interfaces {
        prefix "if";
    }

    import ietf-inet-types {
        prefix inet;
    }

    import ietf-yang-types {
        prefix "yang";
    }

    import ietf-routing-types {
        prefix "rt-types";
    }

import ietf-routing {
    prefix "rt";
}

import ietf-srv6-types {
    prefix srv6-types;
}

organization "IETF SPRING Working Group";
contact
    "WG Web:  <http://tools.ietf.org/wg/spring/>
    WG List:  <mailto:spring@ietf.org>
    Editor:  Kamran Raza
              <mailto:skraza@cisco.com>
    Editor:  Jaganbabu Rajamanickam
              <mailto:jrajaman@cisco.com>
    Editor:  Xufeng Liu
              <mailto:Xufeng_Liu@jabil.com>
    Editor:  Zhibo Hu
              <mailto:huzhibo@huawei.com>
    Editor:  Iftekhar Hussain
              <mailto:IHussain@infinera.com>
    Editor:  Himanshu Shah
              <mailto:hshah@ciena.com>
    Editor:  Daniel Voyer
              <mailto:daniel.voyer@bell.ca>
    Editor:  Hani Elmalky
              <mailto:hani.elmalky@ericsson.com>
    Editor:  Satoru Matsushima
              <mailto:satoru.matsushima@gmail.com>
    Editor:  Katsuhiro Horiba
              <mailto:katsuhiro.horiba@g.softbank.co.jp>
    Editor:  Ahmed AbdelSalam
              <mailto:ahmed.abdelsalam@gssi.it>

";
This YANG module defines the essential elements for the management of Segment-Routing with IPv6 dataplane (SRv6).

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).

reference "RFC XXXX";

revision 2018-03-01 {
  description
    "Updated to align with SRv6 network programming draft rev 04";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
}

revision 2017-11-12 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
}

/*
* Common
*/

grouping path-attrs-cmn {
  description
    "Path properties -common for v4/v6";

  leaf weight {
    type uint32;
    description
      "This value is used to compute a loadshare to perform un-equal load balancing when multiple outgoing path(s) are specified. A share is computed as a ratio of this number to the total under all configured path(s).";
  }

  leaf role {

  }
type enumeration {
    enum PRIMARY { description "Path as primary traffic carrying"; }  
    enum BACKUP { description "Path acts as a backup"; }  
    enum PRIMARY_AND_BACKUP { description "Path acts as primary and backup simultaneously"; }  
}

description "The path role";

leaf backup-path-index {
    type uint8;
    description "Index of the protecting (backup) path";
}

grouping path-out-sids {
    description "Grouping for path’s SID stack";

    list out-sid {
        key "sid";
        description "Out SID";

        leaf sid {
            type srv6-types:srv6-sid;
            description "SID value";
        }
    }
}

grouping path-out-labels {
    description "Grouping for path’s label stack";

    list out-label {
        key "label";
        description "Out label";

        leaf label {
            type rt-types:mpls-label;
            description "Label value";
        }
    }
}

/*
 * Config and State
 */
grouping srv6-encap {
  description "Grouping for encap param config.";
  container encapsulation {
    description "Configure encapsulation related parameters";
    leaf source-address {
      type inet:ipv6-address;
      description "Specify a source address (for T.Encap). The address must logically exist and be routable";
    }
    leaf ip-ttl-propagation {
      type boolean;
      default false;
      description "IP TTL propagation from encapsulated packet to encapsulating outer IPv6 header. When configured on decapsulation side, this refers to propagating IP TTL from outer IPv6 header to inner header after decap";
    }
  }
}

grouping srv6-locator-state {
  description "SRv6 grouping Locastateor ";
  leaf operational-status {
    type srv6-types:srv6-status-type;
    config false;
    description "Indicates whether locator state is UP";
  }
  leaf is-in-address-conflict {
    type boolean;
    config false;
    description "Indicates whether locator address conflicts with some other IPv6 address on the box";
  }
}

grouping srv6-locators {
  description "SRv6 locator grouping";
  container locators {
    description "SRv6 locators";
    list locator {
      key "name";
      description "Configure a SRv6 locator";
    }
  }
}
leaf name {
    type string;
    description "Locator name";
}

leaf enable {
    type boolean;
    default false;
    description "Enable a SRv6 locator";
}

leaf is-default {
    type boolean;
    mandatory true;
    description "Indicates if the locator is a default locator";
}

container prefix {
    description "Specify locator prefix value";
    leaf address {
        type inet:ipv6-address;
        mandatory true;
        description "IPv6 address";
    }
    leaf length {
        type srv6-types:srv6-locator-len;
        mandatory true;
        description "Locator (prefix) length";
    }
}

uses srv6-locator-state;

}

}

grouping srv6-stats-in {
    description "Grouping for inbound stats";

    leaf in-pkts {
        type yang:counter64;
        description "A cumulative counter of the total number of packets received";
    }

    leaf in-octets {
        type yang:counter64;
        description "A cumulative counter of the total bytes received.";
    }
}

grouping srv6-stats-out {
    description "Grouping for inbound stats";
    leaf out-pkts {
        type yang:counter64;
        description "A cumulative counter of the total number of packets transmitted";
    }
    leaf out-octets {
        type yang:counter64;
        description "A cumulative counter of the total bytes transmitted";
    }
}

grouping path-out-sids-choice {
    description "Grouping for Out-SID choices";
    choice encap-type {
        description "Out-SID encap-based choice";
        case srv6 {
            uses path-out-sids;
        }
        case mpls {
            uses path-out-labels;
        }
    }
}

grouping local-sid-fwd-state {
    description "SRv6 local-SID forwarding state grouping";
    container forwarding {
        description "SRv6 local-SID forwarding state";
        leaf is-installed {
            type boolean;
            description "Indicates whether SID is installed in forwarding";
        }
        leaf next-hop-type {
            type srv6-types:srv6-nexthop-type;
            description "Forwarding next-hop types";
        }
    }
}
when ".//is-installed = 'true'" {
    description "This container is valid only when the local-SID is installed in forwarding";
}

list path {
    key path-index;
    description "The list of paths associated with the SID";

    leaf path-index {
        type uint8;
        description "Index of the path";
    }
}

container 12 {
    when ".//..//..//next-hop-type = 'l2'" {
        description "This container is valid only for L2 type of NHs";
    }

    leaf interface {
        type if:interface-ref;
        description "The outgoing Layer2 interface";
    }

    description "L2 information";
}

container 13 {
    when ".//..//..//next-hop-type != 'l2'" {
        description "This container is valid only for L3 type of NHs";
    }

    leaf interface {
        type if:interface-ref;
        description "The outgoing Layer3 interface";
    }

    leaf next-hop {
        type inet:ip-address;
        description "The IP address of the next-hop";
    }

    uses path-attrs-cmn;

    description "L3 information";
}

uses path-out-sids-choice;
description "Forwarding paths";
"
}
"
}

grouping srv6-state-sid {
  description "SRv6 SID state grouping";
  container local-sids {
    config false;
    description "Local-SID state";
    container counters {
      description "SRv6 counters";
      container cnt3 {
        description "Counts SRv6 traffic received/dropped on local prefix not instantiated as local-SID";
        uses srv6-stats-in;
      }
    }
  }
  list local-sid {
    key "sid";
    description "Per-localSID Counters";
    leaf sid {
      type srv6-types:srv6-sid;
      description "Local SID value";
    }
    uses srv6-locator-ref;
    leaf is-reserved {
      type boolean;
      description "Set to true if SID comes from reserved pool";
    }
    leaf end-behavior-type {
      type identityref {
        base srv6-types:srv6-endpoint-type;
      }
      description "Type of SRv6 end behavior.";
    }
    leaf alloc-type {
      type srv6-types:sid-alloc-type;
      description "Type of sid allocation.";
    }
  }
}

list owner {
  key "type instance";
  description "SID Owner clients";
  leaf type {
    type identityref {
      base srv6-types:srv6-sid-owner-type;
    }
    description "SID owner/client type";
  }
  leaf instance {
    type string;
    description "Client instance";
  }
  leaf is-winner {
    type boolean;
    description "Is this client/owner the winning in terms of forwarding";
  }
}

uses local-sid-fwd-state;

counter counters {
  description "SRv6 per local-SID counters";
  container cnt1 {
    description "Counts SRv6 traffic received on local-SID prefix and processed successfully";
    uses srv6-stats-in;
  }
}

grouping srv6-support-ends {
  description "SRv6 End behavior support grouping";
  list end-behavior {
    key "type";
    description "End behavior support";
    leaf type {
      type identityref {
        base srv6-types:srv6-endpoint-type;
      }
      description "End behavior (End*) type";
    }
  }
}
leaf supported {
    type boolean;
    mandatory true;
    description "True if supported";
}
}

grouping srv6-support-transits {
    description "SRv6 Transit behavior support grouping";

    list transit-behavior {
        key "type";
        description "Transit behavior support";

        leaf type {
            type srv6-types:srv6-transit-type;
            description "Transit behavior (T*) type";
        }

        leaf supported {
            type boolean;
            mandatory true;
            description "True if supported";
        }
    }
}

grouping srv6-support-signaled {
    description "SRv6 signaled parameter support grouping";

    container signaled-parameters {
        description "SRv6 signaled parameter support";

        leaf max-sl {
            type uint8;
            //mandatory true;
            description "Maximum value of the SL field in the SRH of
            a received packet before applying the function
            associated with a SID";
        }

        leaf max-end-pop-srh {
            type uint8;
            //mandatory true;
            description "Maximum number of SIDs in the top SRH in an
            SRH stack to which the router can apply
            PSP or USP flavors";
        }

        leaf max-t_insert {

    Raza, et al. Expires September 6, 2018 [Page 37]
type uint8;
//mandatory true;
description "Maximum number of SIDs that can be inserted as part of the T.insert behavior";
}
leaf max-t_encap {
    type uint8;
    //mandatory true;
    description "Maximum number of SIDs that can be inserted as part of the T.Encap behavior";
}
leaf max-end_d {
    type uint8;
    //mandatory true;
    description "Maximum number of SIDs in an SRH when applying End.DX6 and End.DT6 functions";
}
}
}
grouping srv6-support-security-rules {
    description "SRv6 Security rules grouping";
    list security-rule {
        key "type";
        description "Security rule support";
        leaf type {
            type srv6-types:srv6-security-rule-type;
            description "Security rule type";
        }
        leaf supported {
            type boolean;
            mandatory true;
            description "True if supported";
        }
    }
}

grouping srv6-support-counters {
    description "SRv6 Counters grouping";
    list counters {
        key "type";
        description "SRv6 counter support";
        leaf type {
            type srv6-types:srv6-counter-type;
        }
    }
}
description "Counter type";
}
leaf supported {
  type boolean;
  mandatory true;
  description "True if supported";
}
}
}
grouping srv6-state-capabilities {
  description "SRv6 node capabilities grouping";
  container node-capabilities {
    description "Node’s SRv6 capabilities";
    uses srv6-support-ends;
    uses srv6-support-transits;
    uses srv6-support-signaled;
    uses srv6-support-security-rules;
    uses srv6-support-counters;
  }
}

augment "/rt:routing" {
  description "This augments routing-instance configuration with segment-routing SRv6.";
  container srv6 {
    description "Segment Routing with IPv6 dataplane";

    /* config */
    leaf enable {
      type boolean;
      default false;
      description "Enable SRv6";
    }

    uses srv6-encap;
    uses srv6-locators;
    uses srv6-state-capabilities;
    uses srv6-state-sid;
  }
}

/* Notifications */

grouping srv6-locator-ref {
description
"An absolute reference to an SRv6 locator";
leaf locator-ref {
  type leafref {
    path "/rt:routing/srv6:srv6/locators/srv6:locator/srv6:name";
  }
  description
  "Reference to a SRv6 locator.";
}

notification srv6-locator-status-event {
  description
  "Notification event for a change of SRv6 locator operational status.";
  leaf operational-status {
    type srv6-types:srv6-status-type;
    description "Operational status";
  }
  uses srv6-locator-ref;
}

notification srv6-sid-collision-event {
  description
  "Notification event for an SRv6 SID collision - i.e., attempt to bind an already bound SID to a new context";
  leaf sid {
    type srv6-types:srv6-sid;
    description "SRv6 SID";
  }
  container existing {
    description "Current assignment / bind";
    leaf end-behavior-type {
      type identityref {
        base srv6-types:srv6-endpoint-type;
      }
      description "End type";
    }
    // TODO: More
  }
  container requested {
    description "Requested assignment / bind";
    leaf end-behavior-type {
      type identityref {
        base srv6-types:srv6-endpoint-type;
      }
      description "End type";
    }
  }

Raza, et al. Expires September 6, 2018 [Page 40]
6.3. SRv6 Static

 module ietf-srv6-static {

 namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-static";
 prefix srv6-static;

 import ietf-interfaces {
   prefix "if";
 }

 import ietf-inet-types {
   prefix inet;
 }

 import ietf-routing {
   prefix "rt";
 }

 import ietf-srv6-types {
   prefix srv6-types;
 }

 import ietf-srv6-base {
   prefix srv6;
 }

 organization
   "IETF SPRING Working Group";
 contact
   "WG Web:  <http://tools.ietf.org/wg/spring/>
   WG List:  <mailto:spring@ietf.org>"
This YANG module defines the essential elements for the management of Static application for Segment-Routing with IPv6 dataplane (SRv6).

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.
grouping path-attrs-v6 {
  description "IPv6 Path properties";
  leaf interface {
    type if:interface-ref;
    description "The outgoing interface";
  }
  leaf next-hop {
    type inet:ipv6-address;
    description "The IP address of the next-hop";
  }
  uses srv6:path-attrs-cmn;
}

grouping path-attrs-v4 {
  description "IPv4 Path properties";
  leaf interface {
    type if:interface-ref;
    description "The outgoing interface";
  }
leaf next-hop {
    type inet:ipv4-address;
    description "The IP address of the next-hop";
}

uses srv6:path-attrs-cmn;
}

grouping path-attrs-mpls {
    description
        "$MPLS Path properties$";

    leaf interface {
        type if:interface-ref;
        description "The outgoing interface";
    }

    leaf next-hop {
        type inet:ip-address;
        description "The IP address of the next-hop";
    }

    uses srv6:path-attrs-cmn;
}

grouping multi-paths-v6 {
    description "$Multipath grouping$";

    container paths {
        description "List of outgoing paths";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }

            uses path-attrs-v6;
            container encap {
                description "Encapsulation on path";
                uses srv6:path-out-sids;
            }
        }
    }
}
grouping multi-paths-v4 {
  description "Multipath grouping";
  container paths {
    description "List of outgoing paths";
    list path {
      key path-index;
      description "The list of paths associated with the SID";

      leaf path-index {
        type uint8;
        description "Index of the path";
      }

      uses path-attrs-v4;
      container encap {
        description "Encapsulation on path";
        uses srv6:path-out-sids;
      }
    }
  }
}

grouping multi-paths-mpls {
  description "Multipath grouping";
  container paths {
    description "List of outgoing paths";
    list path {
      key path-index;
      description "The list of paths associated with the SID";

      leaf path-index {
        type uint8;
        description "Index of the path";
      }

      uses path-attrs-mpls;
      container encap {
        description "Encapsulation on path";
        uses srv6:path-out-labels;
      }
    }
  }
}

grouping srv6-sid-config {
  description
"Configuration parameters relating to SRv6 sid."

leaf opcode {
  type srv6-types:srv6-func-opcode-unreserved;
  description "SRv6 function opcode."
}

leaf end-behavior-type {
  type identityref {
    base srv6-types:srv6-endpoint-type;
  }
  mandatory true;
  description "Type of SRv6 end behavior."
}

container end {
  when ".../end-behavior-type = 'End'" {
    description "This container is valid only when the user chooses End
        behavior (variant: no PSP, no USP)."
  }
  description "The Endpoint function is the most basic function.
    FIB lookup on updated DA and forward accordingly
    to the matched entry.
    This is the SRv6 instantiation of a Prefix SID
    (variant: no PSP, no USP)"
}

container end_psp {
  when ".../end-behavior-type = 'End_PSP'" {
    description "This container is valid only when the user chooses End
        behavior (variant: PSP only)."
  }
  description "The Endpoint function is the most basic function.
    FIB lookup on updated DA and forward accordingly
    to the matched entry.
    This is the SRv6 instantiation of a Prefix SID
    (variant: PSP only)"
}

container end_usp {
  when ".../end-behavior-type = 'End_USP'" {

container end_psp_usp {
  when "./end-behavior-type = 'End_PSP_USP'" {
    description
    "This container is valid only when the user chooses End
    behavior (variant: PSP/USP).";
  }
}

description
"This container is valid only when the user chooses End
behavior (variant: USP only).";

}

description
"The Endpoint function is the most basic function.
FIB lookup on updated DA and forward accordingly
to the matched entry.
This is the SRv6 instantiation of a Prefix SID
(variant: USP only)";

}

container end-t {
  when "./end-behavior-type = 'End.T'" {
    description
    "This container is valid only when the user chooses
    End.T behavior (variant: no PSP, no USP).";
  }
}

description
"Endpoint with specific IPv6 table lookup (variant: no PSP, no USP).
Lookup the next segment in IPv6 table T
associated with the SID and forward via
the matched table entry.
The End.T is used for multi-table operation
in the core.";

// TODO presence "Mandatory child only if container is present";
leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description
  "Table Id for lookup on updated DA (next segment)";

container end-t_psp {
    when ".../end-behavior-type = 'End.T_PSP'" {
        description
            "This container is valid only when the user chooses End.T behavior (variant: PSP only).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant: PSP only). Lookup the next segment in IPv6 table T associated with the SID and forward via the matched table entry. The End.T is used for multi-table operation in the core.";
        // TODO presence  "Mandatory child only if container is present";
    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}

container end-t_usp {
    when ".../end-behavior-type = 'End.T_USP'" {
        description
            "This container is valid only when the user chooses End.T behavior (variant: USP only).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant: USP only). Lookup the next segment in IPv6 table T associated with the SID and forward via the matched table entry. The End.T is used for multi-table operation in the core.";
        // TODO presence  "Mandatory child only if container is present";
    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}
container end-t_psp_usp {
  when ".../end-behavior-type = 'End.T_PSP_USP'" {
    description
    "This container is valid only when the user chooses
    End.T behavior (variant: USP/PSP).";
  }
}

container end-x {
  when ".../end-behavior-type = 'End.X'" {
    description
    "This container is valid only when the user chooses
    End.X behavior (variant: no USP/PSP)";
  }
}

declaration
"Endpoint with specific IPv6 table lookup (variant: USP/PSP). Lookup the next segment in IPv6 table T associated with the SID and forward via the matched table entry. The End.T is used for multi-table operation in the core.";

// TODO presence "Mandatory child only if container is present";

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description
  "Table Id for lookup on updated DA (next segment)";
}

container end-t_psp_usp {
  when ".../end-behavior-type = 'End.T_PSP_USP'" {
    description
    "This container is valid only when the user chooses
    End.T behavior (variant: USP/PSP).";
  }
}

container end-x {
  when ".../end-behavior-type = 'End.X'" {
    description
    "This container is valid only when the user chooses
    End.X behavior (variant: no USP/PSP)";
  }
}

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description
  "Table Id for lookup on updated DA (next segment)";
}

container end-t_psp_usp {
  when ".../end-behavior-type = 'End.T_PSP_USP'" {
    description
    "This container is valid only when the user chooses
    End.T behavior (variant: USP/PSP).";
  }
}

container end-x {
  when ".../end-behavior-type = 'End.X'" {
    description
    "This container is valid only when the user chooses
    End.X behavior (variant: no USP/PSP)";
  }
}

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description
  "Table Id for lookup on updated DA (next segment)";
}

container end-t_psp_usp {
  when ".../end-behavior-type = 'End.T_PSP_USP'" {
    description
    "This container is valid only when the user chooses
    End.T behavior (variant: USP/PSP).";
  }
}

container end-x {
  when ".../end-behavior-type = 'End.X'" {
    description
    "This container is valid only when the user chooses
    End.X behavior (variant: no USP/PSP)";
  }
}

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description
  "Table Id for lookup on updated DA (next segment)";
}

container end-t_psp_usp {
  when ".../end-behavior-type = 'End.T_PSP_USP'" {
    description
    "This container is valid only when the user chooses
    End.T behavior (variant: USP/PSP).";
  }
}

container end-x {
  when ".../end-behavior-type = 'End.X'" {
    description
    "This container is valid only when the user chooses
    End.X behavior (variant: no USP/PSP)";
  }
}

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description
  "Table Id for lookup on updated DA (next segment)";
}

container end-t_psp_usp {
  when ".../end-behavior-type = 'End.T_PSP_USP'" {
    description
    "This container is valid only when the user chooses
    End.T behavior (variant: USP/PSP).";
  }
}

container end-x {
  when ".../end-behavior-type = 'End.X'" {
    description
    "This container is valid only when the user chooses
    End.X behavior (variant: no USP/PSP)";
  }
}

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description
  "Table Id for lookup on updated DA (next segment)";
}

container end-t_psp_usp {
  when ".../end-behavior-type = 'End.T_PSP_USP'" {
    description
    "This container is valid only when the user chooses
    End.T behavior (variant: USP/PSP).";
  }
}

container end-x {
  when ".../end-behavior-type = 'End.X'" {
    description
    "This container is valid only when the user chooses
    End.X behavior (variant: no USP/PSP)";
  }
}

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description
  "Table Id for lookup on updated DA (next segment)";
}

container end-t_psp_usp {
  when ".../end-behavior-type = 'End.T_PSP_USP'" {
    description
    "This container is valid only when the user chooses
    End.T behavior (variant: USP/PSP).";
  }
}

container end-x {
  when ".../end-behavior-type = 'End.X'" {
    description
    "This container is valid only when the user chooses
    End.X behavior (variant: no USP/PSP)";
  }
}

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description
  "Table Id for lookup on updated DA (next segment)";
}
container end-x_psp {
    when ".../end-behavior-type = 'End.X_PSP'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: PSP only)";
    }

description
    "Endpoint with cross-connect to an array of
    layer-3 adjacencies (variant: PSP only).
    Forward to layer-3 adjacency bound to the SID S.
    The End.X function is required to express any
    traffic-engineering policy.";

leaf protected {
    type boolean;
    default false;
    description "Is Adj-SID protected?";
}

uses multi-paths-v6;
}

container end-x_usp {
    when ".../end-behavior-type = 'End.X_USP'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: USP only)";
    }

description
    "Endpoint with cross-connect to an array of
    layer-3 adjacencies (variant: USP only).
    Forward to layer-3 adjacency bound to the SID S.
    The End.X function is required to express any
    traffic-engineering policy.";

leaf protected {
    type boolean;
    default false;
    description "Is Adj-SID protected?";
}

uses multi-paths-v6;
}

container end-x_psp_usp {
    when ".../end-behavior-type = 'End.X_PSP_USP'" {

Internet-Draft          YANG Data Model for SRv6              March 2018

description
  "This container is valid only when the user chooses
   End.X behavior (variant: PSP/USP)";
}

description
  "Endpoint with cross-connect to an array of
   layer-3 adjacencies (variant: PSP/USP).
   Forward to layer-3 adjacency bound to the SID S.
   The End.X function is required to express any
   traffic-engineering policy."
leaf protected {
  type boolean;
  default false;
  description "Is Adj-SID protected?";
}
uses multi-paths-v6;
}

container end-b6 {
  when ".../end-behavior-type = 'End.B6'" {
    description
      "This container is valid only when the user chooses
       End.B6 behavior.";
  }
  description
    "Endpoint bound to an SRv6 Policy.
     Insert SRH based on the policy and forward the
     packet toward the first hop configured in the policy.
     This is the SRv6 instantiation of a Binding SID.";
  // TODO presence  "Mandatory child only if container is present"
  leaf policy-name {
    type string;
    mandatory true;
    description "SRv6 policy name.";
  }
  uses multi-paths-v6;
}

container end-b6-encaps {
  when ".../end-behavior-type = 'End.B6.Encaps'" {
    description
      "This container is valid only when the user chooses
Description

This is a variation of the End.B6 behavior where the SRv6 Policy also includes an IPv6 Source Address.
Insert SRH based on the policy and update the source IP and forward the packet toward the first hop configured in the policy.
Instead of simply inserting an SRH with the policy (End.B6), this behavior also adds an outer IPv6 header.

// TODO presence "Mandatory child only if container is present"

leaf policy-name {
  type string;
  mandatory true;
  description "SRv6 policy name."
}
leaf source-address {
  type inet:ipv6-address;
  mandatory true;
  description "IPv6 source address for Encap."
}
uses multi-paths-v6;
}

container end-bm {
  when ".../end-behavior-type = 'End.BM'" {
    description "This container is valid only when the user chooses End.BM behavior."
  }
  description "Endpoint bound to an SR-MPLS Policy. Push an MPLS label stack <L1, L2, L3> on the received packet and forward the according to Label L1. This is an SRv6 instantiation of an SR-MPLS Binding SID."
  // TODO presence "Mandatory child only if container is present"
  leaf policy-name {
    type string;
    mandatory true;
  }
}
container end-dx6 {
    when "../../end-behavior-type = 'End.DX6'" {
        description
            "This container is valid only when the user chooses
             End.DX6 behavior.";
    }
    description
        "Endpoint with decapsulation and cross-connect to
         an array of IPv6 adjacencies. Pop the (outer)
         IPv6 header and its extension headers and forward
         to layer-3 adjacency bound to the SID S.
         The End.DX6 used in the L3VPN use-case.";
    uses multi-paths-v6;
}
container end-dx4 {
    when "../../end-behavior-type = 'End.DX4'" {
        description
            "This container is valid only when the user chooses
             End.DX4 behavior.";
    }
    description
        "Endpoint with decapsulation and cross-connect to
         an array of IPv4 adjacencies.
         Pop the (outer) IPv6 header and its extension
         header and forward to layer-3 adjacency bound
         to the SID S.
         This would be equivalent to the per-CE VPN
         label in MPLS.";
    uses multi-paths-v4;
}
container end-dt6 {
    when "../../end-behavior-type = 'End.DT6'" {
        description
            "This container is valid only when the user chooses
             End.DT6 behavior.";
    }
    description
        "Endpoint with decapsulation and specific IPv6 table
         lookup.
         Pop the (outer) IPv6 header and its extension
         headers.";
Lookup the exposed inner IPv6 DA in IPv6
table T and forward via the matched table entry.
End.DT6 function is used in L3VPN use-case."

// TODO presence "Mandatory child only if container is present"

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description "IPv6 table";
}

container end-dt4 {
  when ".../end-behavior-type = 'End.DT4'" {
    description
    "This container is valid only when the user chooses
    End.DT4 behavior."
  }
}

description
"Endpoint with decapsulation and specific
IPv4 table lookup.
Pop the (outer) IPv6 header and its extension
headers.
Lookup the exposed inner IPv4 DA in IPv4
table T and forward via the matched table entry.
This would be equivalent to the per-VRF VPN label
in MPLS.";

// TODO presence "Mandatory child only if container is present"

leaf lookup-table-ipv4 {
  type srv6-types:table-id;
  mandatory true;
  description "IPv4 table";
}

container end-dt46 {
  when ".../end-behavior-type = 'End.DT46'" {
    description
    "This container is valid only when the user chooses
    End.DT46 behavior."
  }
}

description
"Endpoint with decapsulation and specific
IP table lookup.
Depending on the protocol type (IPv4 or IPv6)
of the inner ip packet and the specific VRF name
forward the packet."
This would be equivalent to the per-VRF VPN label in MPLS.

// TODO presence  "Mandatory child only if container is present";

leaf lookup-table-ipv4 {
  type srv6-types:table-id;
  mandatory true;
  description "IPv4 table";
}

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description "IPv6 table";
}

container end-dx2 {
  when "../end-behavior-type = 'End.DX2'" {
    description
      "This container is valid only when the user chooses End.DX2 behavior.";
  }
  description
    "This is an Endpoint with decapsulation and Layer-2 cross-connect to OIF. Pop the (outer) IPv6 header and its extension headers. Forward the resulting frame via OIF associated to the SID. The End.DX2 function is the L2VPN use-case";

  container paths {
    description "List of outgoing paths";

    leaf interface {
      type if:interface-ref;
      mandatory true;
      description "Layer-2 cross-connect to Out interface.";
    }
  }
}

/* TODO
container end-dx2v {
  when "../end-behavior-type = 'End.DX2V'" {
    description
      "This container is valid only when the user chooses End.DX2V behavior.";
  }
  description
    "This container is valid only when the user chooses End.DX2V behavior.";
*/
"Endpoint with decapsulation and specific VLAN L2 table lookup. Pop the (outer) IPv6 header and its extension headers lookup the exposed inner VLANs in L2 table T forward via the matched table entry. The End.DX2V is used for EVPN Flexible cross-connect use-cases";
leaf end-dx2v {
    type empty;
    description "End_DX2V behavior";
}

container end-dt2u {
    when "/../end-behavior-type = 'End.DT2U'" {
        description "This container is valid only when the user chooses End.DT2U behavior.";
    }
    description "Endpoint with decapsulation and specific unicast MAC L2 table lookup. Pop the (outer) IPv6 header and its extension headers. Learn the exposed inner MAC SA in L2 table T. Lookup the exposed inner MAC DA in L2 table T. Forward via the matched T entry else to all L2OIF in T. The End.DT2U is used for EVPN Bridging unicast use cases";
    leaf end-dt2u {
        type empty;
        description "End_DT2U behavior";
    }
}

container end-dt2m {
    when "/../end-behavior-type = 'End.DT2M'" {
        description "This container is valid only when the user chooses End.DT2M behavior.";
    }
    description "Endpoint with decapsulation and specific L2 table flooding. Pop the (outer) IPv6 header and its extension headers. Learn the exposed inner MAC SA in L2 table T. Forward on all L2OIF excluding the one specified in Arg.FE2. The End.DT2M is used for EVPN Bridging BUM use case with ESI filtering capability.";
    leaf end-dt2m {

type empty;

description
  "End_DT2M behavior";
} */
}

container end-otp {
  when "../../end-behavior-type = 'End.OTP'" {
    description
      "This container is valid only when the user chooses
      End.OTP behavior.";
  }
  description
    "Endpoint for OAM with timestamp and punt behavior";
}
}

grouping srv6-static-cfg {
  description
    "Grouping configuration and operation for SRv6 sid.";

  list sid {
    key "opcode";
    description "Local SID list";

    uses srv6-sid-config;
  }
}

augment "../rt:routing/srv6:srv6/locators/srv6:locator" {
  description
    "This augments locator leaf withing SRv6.";

  container static {
    description "Static SRv6";

    /* Local SIDs */
    container local-sids {
      description
        "SRv6-static local-SIDs";

      uses srv6-static-cfg;
      /* no state for now; SID state accessible through base model */
    }
  }
}
7. Security Considerations

The configuration, state, and notification data defined using YANG data models in this document are likely to be accessed via the protocols such as NETCONF [RFC6241] etc.

Hence, YANG implementations MUST comply with the security requirements specified in section 15 of [RFC6020]. Additionally, NETCONF implementations MUST comply with the security requirements specified in sections 2.2, 2.3 and 9 of [RFC6241] as well as section 3.7 of [RFC6536].

8. IANA Considerations

None

9. Acknowledgments

This draft is defining the similar YANG data model as [I-D.hu-spring-srv6-yang]. The authors of that draft have agreed to join this draft.

The authors would like to acknowledge Darren Dukes, Les Ginsberge, and Ahmed Bashandy for their input and review.

10. References

10.1. Normative References

[I-D.filsfils-spring-srv6-network-programming]
10.2. Informative References

[I-D.bashandy-isis-srv6-extensions]


Li, Z., Matsushima, S., and K. Horiba, "YANG Data Model for SRv6", draft-hu-spring-srv6-yang-00 (work in progress), October 2017.


Authors’ Addresses

Kamran Raza
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, ON K2K-3E8
CA

Email: skraza@cisco.com

Jaganbabu Rajamanickam
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, ON K2K-3E8
CA

Email: jrajaman@cisco.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Zhibo Hu
Huawei Technologies

Email: huzhibo@huawei.com

Iftekhar Hussain
Infinera Corporation

Email: IHussain@infinera.com

Himanshu Shah
Ciena Corporation

Email: hshah@ciena.com

Daniel Voyer
Bell Canada

Email: daniel.voyer@bell.ca
Abstract

This document defines data plane functionality required to implement service segments and achieve service chaining in SR-enabled MPLS and IP networks, as described in the Segment Routing architecture.

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-Draft    Segment Routing for Service Chaining        March 2018

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

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction ................................. 3
2. Terminology .................................. 4
3. Classification and steering ................. 5
4. Service Functions ............................ 5
   4.1. SR-aware SFs ............................. 6
   4.2. SR-unaware SFs ........................... 6
5. Service function chaining .................... 7
   5.1. SR-MPLS data plane ....................... 8
      5.1.1. Encoding SFP Information by an MPLS Label Stack ... 8
      5.1.2. Encoding SFC Information by an MPLS Label Stack ... 11
   5.2. SRv6 data plane .......................... 14
      5.2.1. Encoding SFP Information by an SRv6 SRH ......... 14
      5.2.2. Encoding SFC Information by an IPv6 SRH .......... 16
6. SR proxy behaviors ............................ 17
   6.1. Static SR proxy .......................... 20
      6.1.1. SR-MPLS pseudocode ..................... 21
      6.1.2. SRv6 pseudocode ....................... 22
   6.2. Dynamic SR proxy ........................ 25
      6.2.1. SR-MPLS pseudocode ..................... 25
      6.2.2. SRv6 pseudocode ....................... 26
   6.3. Shared memory SR proxy .................. 26
   6.4. Masquerading SR proxy .................... 27
      6.4.1. SRv6 masquerading proxy pseudocode .......... 28
      6.4.2. Variant 1: Destination NAT ............... 28
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, or rich behaviors to support use-cases such as Service Function Chaining (SFC).

In the context of SFC, each Service Function (SF), running either on a physical appliance or in a virtual environment, is associated with a segment, which can then be used in a segment list to steer packets through the SF. Such service segments may be combined together in a segment list to achieve SFC, but also with other types of segments as defined in [I-D.ietf-spring-segment-routing]. SR thus provides a fully integrated solution for SFC, overlay and underlay optimization. Furthermore, the IPv6 dataplane natively supports metadata transportation as part of the SR information attached to the packets.

This document describes how SR enables SFC in a simple and scalable manner, from the segment association to the SF up to the traffic classification and steering into the service chain. Several SR proxy behaviors are also defined to support SR SFC through legacy, SR-unaware, SFs in various circumstances.

The definition of an SR Policy and the steering of traffic into an SR Policy is outside the scope of this document. These aspects are covered in [I-D.filfsfils-spring-segment-routing-policy].
The definition of control plane components, such as service segment discovery, is outside the scope of this data plane document. BGP extensions to support SR-based SFC are proposed in [I-D.dawra-idr-bgp-sr-service-chaining].

Familiarity with the following IETF documents is assumed:

- Segment Routing Architecture [I-D.ietf-spring-segment-routing]
- Segment Routing with MPLS data plane [I-D.ietf-spring-segment-routing-mpls]
- Segment Routing Traffic Engineering Policy [I-D.filsfils-spring-segment-routing-policy]
- Segment Routing Header [I-D.ietf-6man-segment-routing-header]
- SRv6 Network Programming [I-D.filsfils-spring-srv6-network-programming]
- SR-MPLS over IP [I-D.xu-mpls-sr-over-ip]
- Service Function Chaining Architecture [RFC7665]

2. Terminology

This document leverages the terminology introduced in [I-D.ietf-spring-segment-routing], [I-D.filsfils-spring-segment-routing-policy] and [RFC7665]. It also introduces the following new terminology.

SR-aware SF: Service Function fully capable of processing SR traffic

SR-unaware SF: Service Function unable to process SR traffic or behaving incorrectly for such traffic

SR proxy: Proxy handling the SR processing on behalf of an SR-unaware SF

Service Segment: Segment associated with an SF, either directly or via an SR proxy

SR SFC policy: SR policy, as defined in [I-D.filsfils-spring-segment-routing-policy], that includes at least one Service Segment. An SR SFC policy may also contain other types of segments, such as VPN or TE segments.
3. Classification and steering

Classification and steering mechanisms are defined in section 8 of [I-D.filsfils-spring-segment-routing-policy] and are independent from the purpose of the SR policy. From a headend perspective, there is no difference whether a policy contains IGP, BGP, peering, VPN and service segments, or any combination of these.

As documented in the above reference, traffic is classified when entering an SR domain. The SR policy head-end 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, which is associated with a weighted set of segment lists.

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 and SRH encapsulation.

4. Service Functions

A Service Function (SF) may be 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. An SF 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 an SF.

SR enables SFC by assigning a segment identifier, or SID, to each SF and sequencing these service SIDs in a segment list. A service SID may be of local significance or directly reachable from anywhere in the routing domain. The latter 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 SFs 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 SFs. An SR-aware SF 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 SF 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. Any SF that does not meet these criteria is considered as SR-unaware.

4.1. SR-aware SFs

An SR-aware SF is associated with a locally instantiated service segment, which is used to steer traffic through it.

If the SF 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 SF.

If the SF 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 SF-specific functions is out of the scope of this document.

An SRv6-aware SF may also retrieve, store or modify information in the SRH TLVs.

4.2. SR-unaware SFs

An SR-unaware SF 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 SFs in an SR SC policy, it is thus required to remove the SR information as well as any other encapsulation header before the SF receives the packet, or to alter it in such a way that the SF can correctly process the packet.

In this document, we define the concept of an SR proxy as an entity, separate from the SF, that performs these modifications and handle the SR processing on behalf of an SF. The SR proxy can run as a separate process on the SF appliance, on a virtual switch or router
on the compute node or on a remote host. In this document, we only assume that the proxy is connected to the SF via a layer-2 link.

An SR-unaware SF is associated with a service segment instantiated on the SR proxy, which is used to steer traffic through the SF. Section 6 describes several SR proxy behaviors to handle the encapsulation headers and SR information under various circumstances.

5. Service function chaining

When applying a particular Service Function Chain (SFC) [RFC7665] to the traffic selected by a service classifier, the traffic need to be steered through an ordered set of Service Functions (SF) in the network. This ordered set of SFs in the network indicates the Service Function Path (SFP) associated with the above SFC. In order to steer the selected traffic through the required ordered list of SFs, the service classifier needs to attach information to the packet specifying exactly which Service Function Forwarders (SFFs) and which SFs are to be visited by traffic, the SFC, or the partially specified SFP which is in between the former two extremes.

The SR source routing mechanisms can be used to steer traffic through an ordered set of devices (i.e., an explicit path) and instruct those nodes to execute specific operations on the packet.

This section describes how to leverage SR to realize a transport-independent service function chaining by encoding the service function path information or service function chain information as an MPLS label stack or an IPv6 SRH.

```
+-----------------------------------------------------+
|                     SR network                      |
|            +---------+       +---------+            |
|            |   SF1   |       |   SF2   |            |
|            +----+----+       +----+----+            |
|               ^ | |(3)          ^ | |(6)            |
|       (1)  (2)| | V     (4)  (5)| | V     (7)       |
|----+-----+ ---> +----+----+ ----> +----+----+ --->  +----+-----+
|    A     +------+  SFF1   +-------+  SFF2   +-------+    B     |
| Head-end |      |         |       |         |       | Tail-end |
|-----------+      +---------+       +---------+       +------+
Figure 1: Service Function Chaining in SR networks
```

As shown in Figure 1, SFF1 and SFF2 are two SR-capable nodes. They are also SFFs, each with one SF attached. In addition, they have
allocated and advertised segments for their locally attached SFs. For example, SFF1 allocates and advertises a SID (i.e., S(SF1)) for SF1 while SFF2 allocates and advertises a SID (i.e., S(SF2)) for SF2. These SIDs, which are used to indicate SFs, are referred to as service segments, while the SFFs are identified by either node or adjacency segments depending on how strictly the network path needs to be specified. In this example, we assume that the traffic is steered to both SFFs using their node segments S(SFF1) and S(SFF2), respectively.

Now assume that a given traffic flow is steered in an SR policy instantiated on node A with an endpoint B, hereafter referred to as the SR policy head-end and tail-end respectively, and associated with particular SFC requirements (i.e., SF1-> SF2). From an SR policy perspective, SFC is only a particular case of traffic engineering where the SR path includes service functions. An SR-SFC policy inherits all the properties of SR-TE policies as defined in [I-D.filsfils-spring-segment-routing-policy]. Section 5.1 and Section 5.2 describe approaches of leveraging the SR-MPLS and SRv6 mechanisms to realize stateless service function chaining. The complete SFP and SFC information is encoded within an MPLS label stack or an IPv6 SRH carried by the packets, so that no per-chain state is required at the intermediate hops. Since the encoding of the partially specified SFP is just a simple combination of the encoding of the SFP and the encoding of the SFC, this document would not describe how to encode the partially specified SFP anymore.

5.1. SR-MPLS data plane

5.1.1. Encoding SFP Information by an MPLS Label Stack
As shown in Figure 2, the head-end, acting as a service classifier, determines that the selected packet needs to travel through an SFC (SF1->SF2) and steers this packet into the appropriate SR policy as described in [I-D.filsfils-spring-segment-routing-policy]. As a result, the packet is encapsulated with an MPLS label stack containing the segment list <SFF1, SF1, SFF2, SF2, T>. This segment list encodes in a stateless manner the SFP corresponding to the above SFC as an MPLS label stack where each service segment is a local MPLS label allocated from SFFs’ label spaces. To some extent, the MPLS label stack here could be looked as a specific implementation of the SFC encapsulation used for containing the SFP information [RFC7665], which does not require the SFF to maintain per-chain state.
When the encapsulated packet arrives at SFF1, SFF1 knows how to send the packet to SF1 based on the top label (i.e., $S(SF1)$) of the received MPLS packet. We first consider the case where SF1 is an SR-aware SF, i.e., it understands how to process a packet with a prepended SR-MPLS label stack. In this case the packet would be sent to SF1 by SFF1 with the label stack $S(SFF2)\rightarrow S(SF2)$. SF1 would perform the required service function on the received MPLS packet where the payload type is determined using the first nibble of the MPLS payload. After the MPLS packet is returned from SF1, SFF1 would send it to SFF2 according to the top label (i.e., $S(SFF2)$).

If SF1 is an SR-unaware SF, i.e. one that is unable to process the MPLS label stack, the remaining MPLS label stack (i.e., $S(SFF2)\rightarrow S(SF2)$) MUST be stripped from the packet before sending the packet to SF1. When the packet is returned from SF1, SFF1 would re-impose the MPLS label stack which had been previously stripped and then send the packet to SFF2 according to the current top label (i.e., $S(SFF2)$). Proxy mechanisms to support SR-unaware SFs are proposed in section 6 of this document.

When the encapsulated packet arrives at SFF2, SFF2 would perform the similar action to that described above.

By leveraging the SR-MPLS data plane, [I-D.xu-mpls-sr-over-ip] describes a source routing instruction which works across both IPv4 and IPv6 underlays in addition to the MPLS underlay. As shown in Figure 3, if there is no MPLS LSP towards the next node segment (i.e., the next SFF identified by the current top label), the corresponding IP-based tunnel for MPLS (e.g., MPLS-in-IP/GRE tunnel [RFC4023], MPLS-in-UDP tunnel [RFC7510] or MPLS-in-L2TPv3 tunnel [RFC4817]) would be used.
Since the transport (i.e., the underlay) could be IPv4, IPv6 or even MPLS networks, the above approach of encoding the SFP information by an MPLS label stack is fully transport-independent which is one of the major requirements for the SFC encapsulation [RFC7665].

5.1.2. Encoding SFC Information by an MPLS Label Stack

The head-end, acting as a service classifier, determines that the selected packet needs to travel through an SFC (SF1->SF2) and steers this packet into the appropriate SR policy as described in [I-D.filsfils-spring-segment-routing-policy]. This results in the
packet being encapsulated with an MPLS label stack containing the segment list <SF1, SF2, T>, which encodes that SFC. Those SF labels MUST be domain-wide unique MPLS labels. Since it is known to the service classifier that SFF1 is attached with an instance of SF1, the service classifier would therefore send the MPLS encapsulated packet through either an MPLS LSP tunnel or an IP-based tunnel towards SFF1 (as shown in Figure 4 and Figure 5 respectively). When the MPLS encapsulated packet arrives at SFF1, SFF1 would know which SF should be performed according to the current top label (i.e., S(SF1)). Similarly, SFF1 would send the packet returned from SF1 to SFF2 through either an MPLS LSP tunnel or an IP-based tunnel towards SFF2 since it’s known to SFF1 that SFF2 is attached with an instance of SF2. When the encapsulated packet arrives at SFF2, SFF2 would do the similar action as what has been done by SFF1. Since the transport (i.e., the underlay) could be IPv4, IPv6 or even MPLS networks, the above approach of encoding the SFC information by an MPLS label stack is fully transport-independent which is one of the major requirements for the SFC encapsulation [RFC7665].
Figure 4: Packet walk in MPLS underlay
5.2. SRv6 data plane

5.2.1. Encoding SFP Information by an SRv6 SRH
As shown in Figure 6, the head-end, acting as a service classifier, determines that the selected packet needs to travel through an SFC (SF1->SF2) and steers this packet into the appropriate SR policy as described in [I-D.filsfils-spring-segment-routing-policy]. As a result, the packet is encapsulated with an IPv6 header and an SRH containing the segment list <SFF1:SF1, SFF2:SF2, T>. The intermediate segments in this list leverage the SRv6 locator-function concept introduced in [I-D.filsfils-spring-srv6-network-programming] to encode both the SFF and the SF in a single IPv6 SID. The traffic is steered via regular IPv6 forwarding up to the SFF represented in the locator part of the SID and then passed to the SF identified by the SID function. This SRH thus indicates in a stateless manner the SFP corresponding to the above SFC. To some extent, the SRH here could be looked as a specific implementation of the SFC encapsulation used for containing the SFP information [RFC7665], which does not require the SFF to maintain per-chain state.

When the encapsulated packet arrives at SFF1, SFF1 knows how to send the packet to the SF based on the active segment. We first consider the case where SF1 is an SR-aware SF, i.e., it understands how to process an IPv6 encapsulated packet with an SRH. In this case the packet is sent to SF1 by SFF1 with the IP and SR headers (H,SFF2:SF2)(T,SFF2:SF2,SFF1:SF1;SL=0). SF1 performs the required
service function on the received packet, where the payload is determined based on the Next Header field value of last extension header and/or the active segment. After the packet is returned from SF1, SFF1 simply forwards it to SFF2 according to the IPv6 destination address.

If SF1 is an SR-unaware SF, i.e. one that is unable to process IPv6 encapsulated packets with an SRH, the encapsulation headers (i.e., outer IPv6 with any extension header) MUST be stripped from the packet before it is sent to SF1. When the packet is returned from SF1, SFF1 would re-encapsulate the packet with the IPv6 and SR headers that had been previously stripped and then send the packet to SFF2 according to the IPv6 destination address. Proxy mechanisms to support SR-unaware SFs are proposed in section 6 of this document.

When the encapsulated packet arrives at SFF2, SFF2 would perform the similar action to that described above.

5.2.2. Encoding SFC Information by an IPv6 SRH

The head-end, acting as a service classifier, determines that the selected packet needs to travel through an SFC (SF1->SF2) and steers this packet into the appropriate SR policy. This results in the packet being encapsulated with an IPv6 header and an SRH containing the segment list <A1:SF1, A2:SF2, T>. In this case, the locator parts A1 and A2 of the intermediate service segments are anycast prefixes advertised by several SFFs attached to SF1 and SF2, respectively. The policy head-end may thus let the traffic be steered to the closest instance of each SF or add intermediate segments to select a particular SF instance. Furthermore, since it is known to the head-end that SFF1 is attached to an instance of SF1, the encapsulated packet may be sent to SFF1 through an MPLS LSP or an IP-based tunnel. Similar tunneling can then be performed between SFF1 and SF1, and between SFF2 and the tail-end, as illustrated on Figure 7. Since the transport (i.e., the underlay) could be IPv4, IPv6 or even MPLS, the above approach of encoding the SFC information by an IPv6 SRH is fully transport-independent which is one of the major requirements for the SFC encapsulation [RFC7665].
6. SR proxy behaviors

This section describes several SR proxy behaviors designed to enable SR SFC through SR-unaware SFs. A system implementing one of these functions may handle the SR processing on behalf of an SR-unaware SF and allows the SF to properly process the traffic that is steered through it.

An SF 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 SFs at intermediate hops in the segment list. In case an SR-unaware SF 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 SF (MPLS PHP, SRv6 End.D or PSP).

As illustrated on Figure 8, the generic behavior of an SR proxy has two parts. The first part is in charge of passing traffic from the network to the SF. It intercepts the SR traffic destined for the SF via a locally instantiated service segment, modifies it in such a way that it appears as non-SR traffic to the SF, then sends it out on a
given interface, IFACE-OUT, connected to the SF. The second part
receives the traffic coming back from the SF 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 SF and the proxy interface
that receives the traffic coming back from the SF. These can be
physical interfaces or sub-interfaces (VLANs) and, unless otherwise
stated, IFACE-OUT and IFACE-IN can represent the same interface.

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 SF behavior
and the traffic type. It is also possible to use different proxy
mechanisms within the same service chain.
Figure 9: SR proxy summary

Note: The use of a shared memory proxy requires both the SF 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 SF. 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
- **S-ADDR**: Ethernet or IP address of the SF (only for inner type IPv4 and IPv6)
- **IFACE-OUT**: Local interface for sending traffic towards the SF
- **IFACE-IN**: Local interface receiving the traffic coming back from the SF
- **CACHE**: SR information to be attached on the traffic coming back from the SF, 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 SF, 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 SF is to be used as part of a bi-directional SR SC policy. If the proxy and the SF 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 SF. 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 SF, 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 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.

In this scenario, there are no restrictions on the operations that can be performed by the SF 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 SF into an SRv6 policy. However, a static SR proxy segment can be used in only one service chain 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 SF 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.

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. IF payload type is Ethernet THEN
2. Pop all labels
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. IF payload type is IPv4 THEN
2. Pop all labels
3. Forward the exposed packet on IFACE-OUT towards S-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. IF payload type is IPv6 THEN
2. Pop all labels
3. Forward the exposed packet on IFACE-OUT towards S-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
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 S-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 S-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 SF. 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. Therefore, a dynamic proxy segment cannot be the last segment in an SR SC policy. As mentioned at the beginning of Section 6, a different SR behavior should be used if the SF is meant to be the final destination of an SR SC policy.

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 SF.

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 SF 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 between lines 1 and 2 of the static SR-MPLS pseudocode.

1. IF top label S bit is 0 THEN
2. Pop top label
3. IF C(IFACE-IN) different from remaining labels THEN ;; Ref1
4. Copy all remaining labels into C(IFACE-IN) ;; Ref2
5. ELSE
6. Drop the packet

Ref1: 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).
Ref2: 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
2. Decrement SL and update the IPv6 DA with SRH[SL]
3. IF C(IFACE-IN) different from IPv6 encaps THEN       ;; Ref1
4. Copy the IPv6 encaps into C(IFACE-IN)            ;; Ref2
5. ELSE
6. Drop the packet

Ref1: "IPv6 encaps" represents the IPv6 header and any attached extension header.

Ref2: 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 SR proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware SF. This proxy behavior leverages a shared-memory interface with the SF in order to hide the SR information from an SR-unaware SF while keeping it attached to the packet. We assume in this case that the proxy and the SF are running on the same compute node. A typical scenario is an SR-capable vrouter running on a container host and forwarding traffic to virtual SFs 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 SF. 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 SF
- **IFACE-OUT**: Local interface for sending traffic towards the SF
- **IFACE-IN**: Local interface receiving the traffic coming back from the SF

A masquerading SR proxy segment is thus defined for a specific SF 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 SF.

The SF 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 SF simply forwards the packet based on its current Destination Address. In this scenario, we assume that the SF 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 SFs 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 SFs.

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 SF, 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

Ref2: 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

SFs 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

SFs 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

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 (i.e., the NSH which is only used as a metadata container) in an MPLS packet is a potential issue to be addressed. One possible way to address the above issue is: SFFs allocate two different labels for a given SF, one indicates the presence of NSH while the other indicates the absence of NSH. This approach has no change to the current MPLS architecture but it would require more than one label binding for a given SF. Another possible way is to introduce a protocol identifier field within the MPLS packet as described in [I-D.xu-mpls-payload-protocol-identifier].

More details about how to contain metadata within an MPLS packet would be considered in the future version of this draft.

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, [I-D.ietf-6man-segment-routing-header] defines the NSH carrier TLV 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 SF 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 SF and its forwarding plane.
An SR-aware SF 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.

7.2.2. 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 an SFC context, this field can be used to encode basic metadata in the SRH.

8. Implementation status

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.

<table>
<thead>
<tr>
<th></th>
<th>VPP</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Static proxy</td>
<td>Available</td>
</tr>
<tr>
<td>P</td>
<td>Dynamic proxy</td>
<td>In progress</td>
</tr>
<tr>
<td>L</td>
<td>Shared memory proxy</td>
<td>In progress</td>
</tr>
<tr>
<td>P</td>
<td>Static proxy</td>
<td>Available</td>
</tr>
<tr>
<td>S</td>
<td>Dynamic proxy - Inner type Ethernet</td>
<td>In progress</td>
</tr>
<tr>
<td>R</td>
<td>Dynamic proxy - Inner type IPv4</td>
<td>Available</td>
</tr>
<tr>
<td>V</td>
<td>Dynamic proxy - Inner type IPv6</td>
<td>Available</td>
</tr>
<tr>
<td>6</td>
<td>Shared memory proxy</td>
<td>In progress</td>
</tr>
<tr>
<td>S</td>
<td>Masquerading proxy</td>
<td>Available</td>
</tr>
<tr>
<td>R</td>
<td>Masquerading proxy - NAT variant</td>
<td>In progress</td>
</tr>
<tr>
<td>V</td>
<td>Masquerading proxy - Cache variant</td>
<td>In progress</td>
</tr>
</tbody>
</table>

Open-source implementation status table
9. Related works

The Segment Routing solution addresses a wide problem that covers both topological and service chaining 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.farrel-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:

<table>
<thead>
<tr>
<th>Value/Range</th>
<th>Hex</th>
<th>Endpoint function</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA</td>
<td>TBA</td>
<td>End.AN - SR-aware function (native)</td>
<td>[This.ID]</td>
</tr>
<tr>
<td>TBA</td>
<td>TBA</td>
<td>End.AS - Static proxy</td>
<td>[This.ID]</td>
</tr>
<tr>
<td>TBA</td>
<td>TBA</td>
<td>End.AD - Dynamic proxy</td>
<td>[This.ID]</td>
</tr>
<tr>
<td>TBA</td>
<td>TBA</td>
<td>End.AM - Masquerading proxy</td>
<td>[This.ID]</td>
</tr>
</tbody>
</table>

Table 1: SRv6 SFC Endpoint Types

11. Security Considerations

The security requirements and mechanisms described in [I-D.ietf-spring-segment-routing] and [I-D.ietf-6man-segment-routing-header] also apply to this document.

Furthermore, it is fundamental to the SFC design that the classifier is a trusted resource which determines the processing that the packet will be subject to, including for example the firewall. Where an SF is not SR-aware the packet may exist as an IP packet, however this is an intrinsic part of the SFC design which needs to define how a packet is protected in that environment. Where a tunnel is used to link two non-MPLS domains, the tunnel design needs to specify how it is secured.
Thus the security vulnerabilities are addressed in the underlying technologies used by this design, which itself does not introduce any new security vulnerabilities.

12. Acknowledgements

The authors would like to thank 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 (Cisco), S. Bryant (Huawei), H. Assarpour (Broadcom), H. Shah (Ciena), L. Contreras (Telefonica I+D), J. Tantsura (Individual), M. Vigoureux (Nokia) and J. Bhattacharyya (Cisco) substantially contributed to the content of this document.

14. References

14.1. Normative References


14.2. Informative References


[I-D.xu-mpls-sr-over-ip]
Xu, X., Bryant, S., Farrel, A., Bashandy, A., Henderickx, W., and Z. Li, "SR-MPLS over IP", draft-xu-mpls-sr-over-ip-00 (work in progress), February 2018.


Authors’ Addresses

Francois Clad (editor)
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Xiaohu Xu (editor)
Alibaba

Email: xiaohu.xxh@alibaba-inc.com
Stefano Salsano
Università di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it